NUMERO 100

ioPROGRAMMO

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • MARZO 2006 • ANNO X, N.3 (100)

VERSIONE PLUS RIVISTA+LIBRO+CD €9,90

VERSIONE STANDARD RIVISTA+CD €6,90

XAML: IL NUOVO LINGUAGGIO MICROSOFT PER SCRIVERE APPLICAZIONI DESKTOP IN MODO FACILE E INTUITIVO

DAI UNA VOCE A VISUAL BASIC

Scrivi un'applicazione che legge un testo con l'accento e i toni di una persona

TEORIA Come funzionano i metodi di sintesi della voce. Evitiamo l'effetto robot

TECNICA La guida passo passo per usare subito e facilmente le Microsoft Speech Api

PRATICA Il codice d'esempio per realizzare velocemente un tuo programma

CASI DI STUDIO: ASP.NET

WEB REVOLUTION

Usa Ajax e converti le tue applicazioni alla nuova tecnologia che evita la lentezza nel refresh delle pagine

VISUAL BASIC

DA ACCESS A SQL SERVER 2005

Trasforma i tuoi programmi per farli lavorare con il "SuperDatabase" di Microsoft

JAVA

XML FACILE CON JAXB

Crea automaticamente le classi per usare i dati che ti servono

ALGORITMI

IL MASSIMO COMUN DIVISORE

Impariamo come usare la ricorsione e gli altri metodi con un esempio divertente

100 SCOPRI IL CODICE E VINCI

Cerca nelle pagine interne la stringa nascosta e vinci fantastici premi

AUTORADIO CON LETTORE MP3

WEBCAM USB

TASTIERA E MOUSE WIRELESS

PAG. 57

IOPROGRAMMO BY EXAMPLE

Gli esempi guidati per imparare un linguaggio in modo pratico e divertente

C#, Visual Basic.NET
Come posso aggiungere uno SplashScreen alla mia applicazione?

PHP
Come posso creare immagini dagli angoli arrotondati, dinamicamente?

Java
Come posso leggere un file XML con DOM?

JavaScript
Come posso fare eseguire un suono al passaggio del mouse su un link?

E tanti altri all'interno...

JAVA

FATTI IL FANTACALCIO CON SPRING

Scopri il framework che li unisce tutti! Un solo strumento per ogni esigenza

IO PROGRAMMO TU DISEGNI

Alla scoperta di Velocity, il tool che separa l'interfaccia dal codice

SQL

UNA SOLA QUERY TANTI RISULTATI

Per filtrare i dati in molti modi senza dover riscrivere il codice

PHP

PHP E XML

Impariamo come prendere i dati da un file oppure creare un nostro archivio

Anno X - N.ro 3 (100) - Marzo 2006 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: M. Autiero, M. Bigatti, L. Buono, F. Grimaldi, F. Lippo, A. Lacava, M. Locuratolo, A. Pelleriti, R. Sidoti, F. Smelzo.

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831213 - Fax 02 83121330
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti

ABBONAMENTO E ARRETRATI
ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90
Offerte valide fino al 31/03/06
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it
Servizio Abbonati:
tel. 02 831212
e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Febbraio 2006

100 di questi numeri

E così siamo giunti a 100! E non lo gridiamo perché sia un merito ottenuto con fatica, ma solo perché ogni cambio epocale si festeggia fra amici. E ioProgrammo e i suoi lettori, possiamo dirlo sono una famiglia di amici. Da quasi dieci anni condividiamo la stessa passione: l'informatica. E si, perché noi programmatori rappresentiamo un po' l'elite del mondo informatico, coloro che ne comprendono i meccanismi più profondi, coloro che sanno sempre cosa si nasconde dietro a ogni click del mouse. ioProgrammo si può vantare di avere contribuito a formare questa enorme famiglia di amanti dell'informatica, anzi di più di esserne parte. Nel corso di questi 100 numeri abbiamo visto variare tendenze e tecnologie, possiamo dire di avere attraversato un'era. Qualcuno ci accusato di avere reso troppo facile il mondo dello sviluppo, troppo accessibile e perciò troppo inflazionato. Noi di questa accusa preferiamo vantarci, perché la conoscenza è un bene condiviso, la diffusione dell'informazione consente di migliorare la qualità della vita. E così il nostro contributo ha reso possibile nel tempo, lo diciamo con l'umiltà di chi è consapevole di essere parte di un movimento giganteso, creare dei programmatori migliori e questo non può che far bene al mondo dello sviluppo. Certo, il merito non è tutto nostro, informarsi continuamente su Internet, tramite corsi, leggendo è certamente un m odo per diventare dei professionisti migliori ed alimentare contemporaneamente le proprie passioni. Tuttavia, ci siamo anche noi, a parlarvi delle nuove tecnologie, ad illustrarvi con parole sempre semplici le soluzioni a questo o quel problema, ci siamo anche noi, nonostante 100 numeri sulle spalle, siamo sempre freschi, pronti a supportarvi nel vostro lavoro giornaliero, pronti a ricercare per voi le informazioni più aggiornate, pronti a fornirvi le u tility migliori. Ci siamo oggi e contiamo di esserci insieme a voi quando ci troveremo di nuovo qui a festeggiare il prossimo traguardo.

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre \soft\codice\ e \soft\tools\) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

DAI UNA VOCE A VISUAL BASIC

Scrivi un'applicazione che legge un testo con l'accento e i toni di una persona umana

- ✔ Teoria: come funzionano i metodi di sintesi della voce
- ✔ tecnica: le dritte per usare le Microsoft Speech Api
- ✔ pratica: il codice che ti detta una ricetta mentre tu cucini

pag. 14

WEB REVOLUTION

QUALCHE CONSIGLIO UTILE

I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

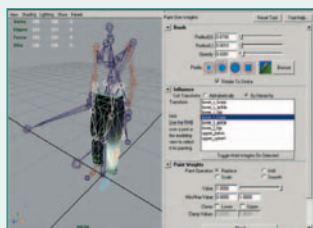
http://forum.ioprogrammo.it

Prodotti del mese

ANIMADEAD 2.0

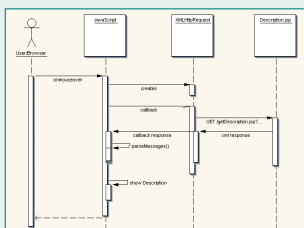
Per creare animazioni partendo da uno scheletro

Una libreria che sfrutta il concetto di skeletal animation.
Si inizia disegnando con un qualunque ambiente 3D lo scheletro di un soggetto, su questo scheletro si costruiscono i movimenti che saranno pilotati dalla libreria.
Si tratta di un progetto per alcuni versi ancora embrionale, ma che lascia intravedere un approccio piuttosto innovativo all'animazione di modelli tridimensionali. Per il disegno dello skeleton si possono attualmente utilizzare modelli progettati attraverso Maya, sono allo studio importer per altri tipi di software.
In ogni caso i risultati sono eccellenti, la qualità dell'animazione è elevata e soprattutto i file utilizzati ammontano a pochi kappa di spazio. In generale si tratta di una libreria anche se non pienamente matura, da tenere d'occhio. La disponibilità del sorgente è una sicurezza in termini di sviluppo.

[pag.105]

Ajax.net

Il componente per usare Ajax con Visual Studio

Di Ajax continuiamo a parlarne in questo numero con il bell'articolo di Fabio Cozzolino. Ajax è una di quelle tecnologie che sta radicalmente cambiando il modo di programmare il Web. Sostanzialmente consente di effettuare un reload di piccoli pezzi di pagina senza dover ricaricare l'intera pagina. È concettualmente diverso da un iFrame ad esempio, perché non si tratta di un componente separato dalla pagina stessa, ma di un elemento incorporato che viene modificato a runtime sulla base della tecnologia Ajax, del DOM e di JavaScript. Tutto questo consente di velocizzare enormemente il tempo di load di una pagina web, e di creare applicazioni internet, la cui interfaccia ha un comportamento simile a quella delle applicazioni standalone, con gli innegabili vantaggi del caso.

[pag.106]

Tutos 1.2.2

Il leader dei software di WorkGroup

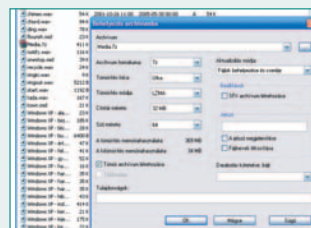
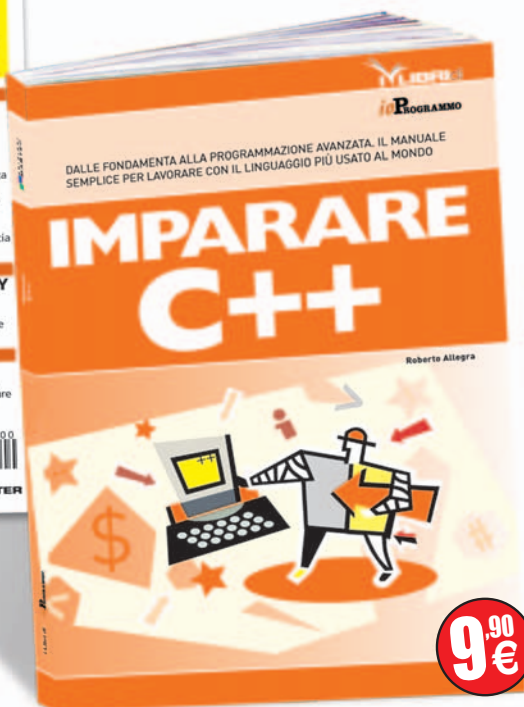
Se vi trovate a dover gestire un gruppo di lavoro o un progetto o più gruppi che lavorano su più progetti, tutos è il software che fa per voi. Si tratta di una web application che consente di automatizzare le scadenze ,lo stato d'esecuzione, le risorse, tutti i parametri tipici necessari all'organizzazione delle procedure gestionali che consentono di ottimizzare il flusso del lavoro. Tutos esporta una serie di funzionalità fuori dal comune che vanno dalla creazione dei diarammi di Gantt fino alla gestione e alla condivisione dell'indirizzario. Assolutamente un must per chi si occupa di gestione della produzione, con i vantaggi tipici della condivisione via rete di informazioni essenziali che coinvolgono l'azienda a più livelli.

[pag.107]

Sevenzip 4.3.2

Un software di compressione eccezionale

All'inizio fu Winzip, poi venne l'era dei cloni! SevenZIP ha superato l'illustre progenitore. L'algoritmo di compressione utilizzato da SevenZip in molti casi fornisce risultati superiori a qualunque aspettativa.
L'algoritmo di compressione di SevenZip è piuttosto interessante per coloro che sono affascinati dalla potenza pure. SevenZip è completamente OpenSource, sono disponibili i sorgenti e questo rappresenta un'ottima occasione per i più curiosi di capire come funzionano gli algoritmi che stanno alla base delle tecniche di compressione. Tuttavia rimane il vantaggio di disporre di un'applicazione realmente utile che in molti casi si può bellamente sostituire a software commerciali di grido.

[pag.107]

I contenuti del libro

Imparare C++

Il c++ è il padre di ogni linguaggio di programmazione. L'eleganza, la portabilità, la flessibilità ne fanno uno strumento in grado di controllare ogni aspetto del ciclo di vita di un'applicazione. Il fascino che questo linguaggio esercita su ogni programmatore è dovuto principalmente alla sua assoluta capacità di essere controllabile in ogni elemento. Tanta flessibilità induce a muoversi attraverso terreni inesplorati il cui unico limite è la fantasia. Viceversa, tanta potenza necessita di un'assoluta padronanza dei meccanismi di base così come di quelli avanzati. Roberto Allegra si muove agevolmente attraverso un universo tanto sconfinato quanto affascinante, illustrando gli aspetti di base ma arrivando a trattare anche argomenti complessi utili per i programmatori più smaliziati.

Dalle fondamenta alla programmazione avanzata. Il manuale semplice per lavorare con il linguaggio più usato al mondo

- **Le basi del linguaggio ed i costrutti fondamentali**
- **Il controllo del flusso e i costrutti diselezione**
- **Lavorare con un linguaggio tipizzato**
- **Programmazione procedurale e ad oggett**

News

UN PEZZO DI WINDOWS RESO DISPONIBILE

Siamo dunque all'ennesimo episodio che vede coinvolti l'Unione Europea e Microsoft. L'azienda di Bill Gates era stata condannata già più di due anni fa con l'accusa di abuso di posizione dominante. Il perno su cui l'accusa faceva leva era l'assenza di sufficienti informazioni che consentissero a terze parti di sviluppare applicazioni perfettamente compatibili con Microsoft Windows. Proprio in questi giorni il gigante di Redmond ha inteso regolarizzare la propria posizione, rendendo disponibile una parte del codice di Windows Desktop e di Windows Server. In particolare il sottosistema che si occupa di gestire la comunicazione fra il kernel e i vari moduli. La diffusione del codice seguirà ancora una volta un modello a licenze. Di fatto per poter dare una sbirciatina al codice sorgente di Windows sarà necessario comprare un'apposita licenza. A questo punto la questione torna nelle mani della UE che dovrà valutare se i passi di Microsoft possono ritenersi sufficienti.

ARRIVA IL PRIMO CLONE DI FLASH

A dire il vero non si tratta di un clone dell'intero ambiente di programmazione, più semplicemente si tratta di un clone OpenSource del player di Flash e che prende il nome di Gnash. L'annuncio ha lasciato del tutto indifferente Adobe che è ora l'attuale proprietaria di Macromedia. Ha invece destato un certo interesse nel mondo OpenSource che ha aggiunto un ulteriore tassellino nella disponibilità del codice aperto. L'aspetto più interessante sembrerebbe riguardare il modo in cui il clone è stato creato. Di fatto nessun aiuto sarebbe giunto da Adobe e nessuna tecnica di reverse enginering sarebbe stata adottata. Questo non può essere che la misura di quanto la community OpenSource sia matura e disponga di menti in grado di competere con le grandi multinazionali del software.

LINUS TORVALDS BOCCIA LA GPL3

Che siate degli estimatori del movimento OpenSource o che non lo siate, non è possibile negare quanto il software libero e la licenza che ne regolamenta l'uso abbia influito e stia ancora influendo sul mondo dell'informatica in generale e sulla programmazione in particolare. Sono soprattutto i programmatori, infatti, a scegliere di diffondere le loro opere coprendone i diritti con la licenza GPL oppure no. In tal senso Linus Torvalds può essere considerato come il programmatore che ha creato il software OpenSource più diffuso al mondo: il kernel di Linux. Le dichiarazioni di Torvalds rispetto all'adozione della GPL3 assumono pertanto un'importanza rilevante, e Torvalds in un suo recente post ha bocciato senza appello la GPL3. Le obiezioni maggiori riguardano alcuni punti fortemente voluti da Richard Stallman, il padre della GPL e il filosofo/pensatore del movimen-

L'AMERICA CHIEDE AIUTO A GOOGLE

Che la questione della privacy in campo informatico sia un problema spinoso lo dimostrano ogni giorno i timori di vedere intercettati i propri log oppure la trasmissione dei propri dati. Suscita una certa preoccupazione che un governo possa utilizzare Internet come una sorta di occhio elettronico puntato sui nostri movimenti. D'altra parte, è recente la notizia secondo la quale il governo americano nel tentativo di combattere i movimenti pedopornografici che infestano il paese abbia chiesto a Google di poter accedere a parte dei database dell'enorme motore di ricerca. L'idea sarebbe quella di analizzare le query che vengono effettuate dagli utenti, per capire come i pedofili si muovono sulla rete attraverso l'uso di parole chiave ben determinate. Sembrerebbe che Google abbia fin qui negato al governo l'accesso ai propri archivi, tuttavia il contenzioso legale rimane aperto. Soprattutto rimane aperto un contenzioso morale, ovvero quanto è corretto che informazioni private del genere vengano scandagliate, se pur per un nobile scopo? D'altra parte questo mette ancora in rilievo, come se fosse ulteriormente necessario, la questione relativa al controllo dei diritti tramite DRM sui calcolatori informatici. Le grandi aziende produttrici di software rappresentano una risorsa, ma anche un pericolo.

to. Il pomo della discordia risiederebbe soprattutto nel netto rifiuto posto dalla GPL3 all'avvento del DRM, la nascente tecnologia che tanto sta facendo discutere il mondo dell'informatica. Torvalds già in tempi non sospetti aveva dichiarato che il DRM è compatibile con Linux e che non avrebbe avuto difficoltà nel gestirlo all'interno del suo kernel.
Immediatamente a seguire il padre del pinguino ha specificato di non volere abbandonare la GPL, piuttosto semplicemente ritiene di voler adottare la versione della GPL fin qui utilizzata per il proprio Kernel. D'altra parte non ci sarebbe niente di male nello specificare, all'interno dei vari software, la versione della GPL a cui fanno riferimento. Si apre dunque una nuova era dell'OpenSource, speriamo che un'eventuale spaccatura non incida su un movimento che fino ad ora ha fatto proprio dell'unità di intenti e del sentire comune la sua forza maggiore.

TAIWAN ABBANDONA MICROSOFT

Se considerate che una consistente fetta delle apparecchiature hardware che popolano le nostre scrivanie recano la scritta "Made in Taiwan", potete ben comprendere quanto la posizione occupata dalla potente nazione Asiatica sia importante per il mercato informatico. Ed è proprio da Taiwan che arriva una battuta d'arresto per il gigante di Redmond. Il parlamento taiwanese ha infatti appena decretato un taglio del 25% alle spese effettuate per l'acquisto di software targato Microsoft. Secondo gli analisti, l'obiettivo del governo Taiwanese sarebbe duplice. Si intende sia avvertire Microsoft che il paese asiatico non è disposto a subire una posizione di predominanza da una qualsiasi azienda informatica, sia diminuire le spese complessive adottate dalla pubblica amministrazione per l'acquisto di prodotti targati MS.

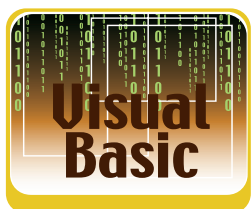
SUL WMF NON CI SONO DUBBI IL FORMATO E' SICURO

I bug sul formato WMF hanno afflitto Windows fin dalle prime versioni, tuttavia MS sembra avere sempre negato la possibilità che i propri sistemi possano essere violati sfruttando una vulnerabilità del noto formato. L'ennesima risposta ai dubbi espressi da gran parte degli utilizzatori è giunta qualche giorno fa, con la solita affermazione che si presta a più di un'interpretazione. Il WMF sarebbe sicuro, limitatamente al fatto che effettuare un exploit su questo formato comporterebbe una violazione del formato stesso, e che il bug introdotto in WMF non si può considerare come una backdoor lasciata intenzionalmente aperta. Fin qui le dichiarazioni di Microsoft, la questione comunque non ha convinto fino in fondo gli sviluppatori, che restano scettici.

UN VIRUS WRITER ITALIANO ALLA SBARRA

Accesso abusivo a sistemi informatici, questa la motivazione con cui è stato condannato a mesi sei di reclusione, sostituiti con 6800 euro di ammenda, l'autore di Vierika, un virus diffusosi nel 2001 sui sistemi dotati di Microsoft Outlook. La sentenza ha suscitato parecchi dubbi in relazione sia alla motivazione che alle modalità con cui l'indagine è stata condotta. Da un lato infatti si contesta l'effettiva pericolosità del Virus, un codice scritto in Visual Basic, non complesso. L'accusa di "Accesso abusivo a sistemi informatici" si configura come plausibile ma non esattamente corrispondente al danno arrecato. D'altro canto si contestano le modalità con cui sono state reperite le informazioni che hanno condotto all'arresto del giovane. Ricerche effettuate con procedure ancora non ben determinate e quindi facilmente contestabili. Tutta la vicenda ha messo in luce due aspetti. Il primo riguardante come si stia pericolosamente diffondendo questo tipo di criminalità tale da diventare una minaccia per aziende di ogni tipo che rimangono esposte al probabile ricatto monetario al fine di non essere sottoposte al tiro degli hacker. Il secondo come la normativa italiana non disponga ancora di procedure certe per accertare questo genere di crimine.

VISUAL BASIC TE LE RACCONTA

LE SPEECH API (SAPI) E, PERMETTONO LO SVILUPPO DI APPLICAZIONI BASATE SULLA VOCE IN PARTICOLARE SUL RICONOSCIMENTO E SULLA SINTESI VOCALE, OVVERO LA TRASFORMAZIONE DI TESTO IN PARLATO. VEDIAMO COME USARLE....

Grazie alle Speech API (SAPI), è possibile sviluppare applicazioni basate sulla voce, in particolare sul riconoscimento vocale e sulla sintesi vocale, cioè la conversione di testo in parlato, nota come Text-To-Speech (TTS).
In questo articolo realizzeremo, un'applicazione che utilizzi Microsoft Speech SDK 5.1 e VB.NET 2003, per leggerci le ricette mentre siamo alle prese con i fornelli. Tutte le Speech Api sono accessibili in VB.NET attraverso il modello ad oggetti COM (Component Object Model).

SISTEMI DI SINTESI

Un sistema di sintesi della voce è una macchina che, a partire da appropriate rappresentazioni di un evento linguistico (parola, messaggio, insiemi di messaggi, ecc.), è in grado di produrre l'emissione sonora corrispondente. La sintesi della voce consente ad un computer di inviare istruzioni o informazioni all'utente attraverso il parlato.
Alcuni vantaggi di questa tecnologia sono:

- chiunque può facilmente comprendere il messaggio senza addestramento o intensa concentrazione.
- il messaggio può essere ricevuto anche quando l'ascoltatore è coinvolto in altre attività, come camminare, guidare, muovere oggetti o osservare qualcosa.
- la convenzionale linea telefonica può essere usata per realizzare facili accessi a informazioni remote.

I metodi di sintesi della voce possono essere suddivisi in due categorie:

- Sintesi basata sulla codifica di singole parole di voce umana registrata. Tali frammenti sono opportunamente combinati per la composizione del messaggio parlato (sistemi a vocabolario limitato).
- Sintesi basata su regole linguistiche, fonetiche e acustiche (sistemi a vocabolario illimitato).

I sistemi di sintesi a "vocabolario limitato" possono riprodurre, come indica la denominazione, un numero limitato, seppure grande, di parole e/o messaggi.
Nel primo caso, la tecnologia impiegata è molto semplice, la macchina attua la gestione di un archivio che contiene la registrazione degli eventi linguistici da riprodurre. Il messaggio viene poi generato selezionando i vocaboli , giustapponendo, attraverso regole opportune, i corrispondenti parametri, inviandoli ad un circuito, che dà luogo all'emissione sonora desiderata.
Con i sistemi di sintesi da testo a "vocabolario illimitato", è possibile trasformare in messaggio verbale un qualunque testo. Per l'elaborazione del segnale sono impiegati diversi stadi di elaborazione basati su differenti competenze specialistiche: linguistiche, fonetiche, acustiche.
Naturalmente le SAPI implementano il metodo di sintesi a "vocabolario illimitato".

SINTESI VOCALE CON L'INTERFACCIA SPVOICE

Per accedere alle Speech Api, è necessario selezionare, dall'ambiente di sviluppo di VB.Net, la voce di menu Progetto/Aggiungi Riferimento. Dalla finestra di dialogo dobbiamo aprire la scheda COM, e selezionare la voce Microsoft Speech Object Library, come viene mostrato nell'immagine seguente. In questo modo avremo a disposizione quanto

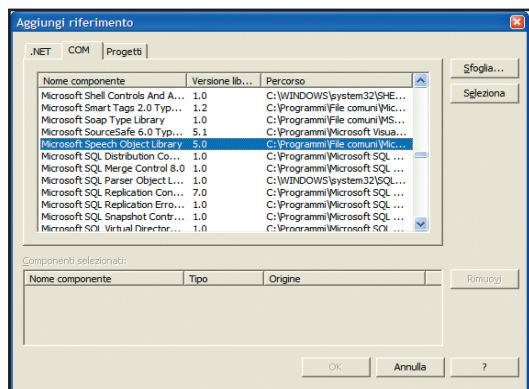



Fig. 1: ***La finestra di dialogo che consente di aggiungere la Microsoft Speech Library al progetto***

ci serve per realizzare i nostri scopi.
In tutti gli esempi di codice, per evitare di scrivere ogni volta il nome completo della libreria, diamo per scontato l'inserimento della seguente istruzione Imports:

```
Imports SpeechLib
```

L'interfaccia SPVoice rende possibile la conversione di un testo qualsiasi in parlato. L'uso dell'interfaccia SPVoice è molto semplice, infatti per consentire al Computer, di pronunciare una frase qualsiasi, è sufficiente chiamare il metodo Speak, passando come parametro la frase che deve essere pronunciata.
Il codice necessario per pronunciare il classico "Ciao Mondo" sarà semplicemente:

```
Dim Voice As SpVoice
Voice = New SpVoice
Voice.Speak("Ciao Mondo")
```

Il metodo Speak, ammette un parametro opzionale, che permette, tra l'altro, di impostare il modello di programmazione. Per default il modello di programmazione è sincrono, cioè il controllo del programma non passa all'istruzione successiva fino a quando non termina la sintesi del testo.
Quindi se riscriviamo il codice precedente in:

```
Dim Voice As SpVoice
Voice = New SpVoice
Voice.Speak("Ciao Mondo")
MessageBox.Show("Fine Lettura")
```

Il messaggio "Fine Lettura" apparirà a video soltanto dopo che il testo sia stato letto per intero.
Se vogliamo evitare questo effetto dovremo passare il valore SpeechVoiceSpeakFlags.SVSFlagsAsync come secondo argomento del metodo Speak.

Il codice seguente mostra un esempio d'uso della sintassi appena esposta:

```
Dim Voice As SpVoice
Voice = New SpVoice
Voice.Speak("Ciao Mondo",
              SpeechVoiceSpeakFlags.SVSFlagsAsync)
MessageBox.Show("Fine Lettura")
```

In questo modo il controllo del programma passa subito all'istruzione successiva ed il messaggio apparirà a video prima che sia terminata la lettura della stringa di testo. Un altro valore interessante, che utilizzeremo in seguito, è SVSFPurgeBeforeSpeak che elimina tutte le voci pendenti.

PROCESSI FONDAMENTALI DEL PROCESSORE LINGUISTICO.

A questo punto non possiamo esimerci dal passare alla teoria per descrivere i processi fondamentali del processore linguistico di un sistema TTS, in modo da comprendere le difficoltà legate alla corretta pronuncia di una frase, ed avere un quadro completo della complessità della sintesi vocale.

NORMALIZZAZIONE DEL TESTO

In generale, un testo scritto, contiene del materiale simbolico quali numeri, abbreviazioni, acronimi che saranno interpretate appropriatamente da un lettore umano usando sia una analisi contestuale che conoscenze di tipo linguistico, culturale ecc. Ad esempio,

INGLESE O ITALIANO?

Gli SE inclusi nel pacchetto full delle SAPI sono tutti in versione inglese ma nonostante tutto leggono discretamente anche l'italiano, in ogni caso la programmazione del TTS non dipende dallo SE usato e quindi dalla lingua usata, per cui, basta installare uno SE italiano per avere una corretta lettura del testo nella nostra lingua.

BREVE STORIA DELLA SINTESI VOCALE

Fino dall'antichità esistono leggende su automi capaci di comunicare con l'uomo attraverso la voce, la più antica narra di una statua del dio Memnon (XV secolo A.C.) che levava un canto mattutino in onore di sua madre EOS. Verso il 1880 fece enorme scalpore un automa esposto a Londra fornito di un mantice e di una tastiera che era in grado di pronunciare tutte le parole. Ma a parte queste curiosità scientifiche, la data di nascita dei moderni sistemi in grado di parlare si può fissare al 1939 quando viene presentato il VODER (Voice Operation DEmonstratoR). Successivamente negli anni '50 si arrivò per la prima volta alla definizione del primo modello elettronico, seppure analogico, del tratto vocale. Il riconoscimento del parlato è invece di più moderna realizzazione, si deve aspettare infatti il 1950 per avere il primo rudimentale riconoscitore del parlato.

la sequenza numerica 10.12.90 può indicare una data, un numero telefonico o un qualsiasi altro codice numerico: solo una analisi del contesto semantico permette di selezionare il significato più appropriato. Allo stesso modo, alcuni simboli grafici quali ".", ",", ";" ecc., possono servire come separatori ad esempio in una sequenza di numeri rappresentanti una cifra (123.000 centoventitremila) oppure possono rappresentare dei simboli di punteggiatura che concorrono alla determinazione della pronuncia di una frase (stress, durata, intonazione, pause ecc.).
Il processo di normalizzazione ha come fine ultimo, l'eliminazione di queste ambiguità attraverso una trasformazione dei simboli utilizzati nel testo in parole: ad esempio Dott. sarà trasformato in dottore, 1990 in millenovecentonovanta oppure uno nove nove zero, ecc.

SCARICARE LE API

La versione completa dello Speech sdk 5.1, può essere scaricato dal sito della Microsoft all'indirizzo http://www.microsoft.com/speech/download/sdk51/

ASSEGNAMENTO DELLO STRESS (ACCENTO)

La determinazione della corretta posizione dello stress lessicale assume un ruolo centrale nei sistemi TTS. Esso gioca infatti un ruolo fondamentale non solo per la pronuncia corretta della singola parola, ma anche per la determinazione della intonazione corretta e del ritmo di una intera frase.
Questo problema è amplificato nel caso della lingua italiana, caratterizzata dal fatto che l'accento tonico può trovarsi in qualsiasi posizione dalla prima all'ultima sillaba di una parola, e solo le parole che terminano con una vocale tonica, ad es. città, richiedono l'uso del segno grafico per indicare lo stress.

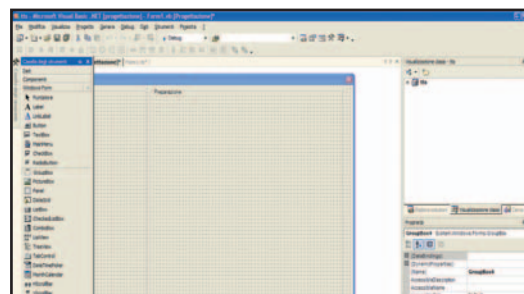
CONVERSIONE IN FONEMI

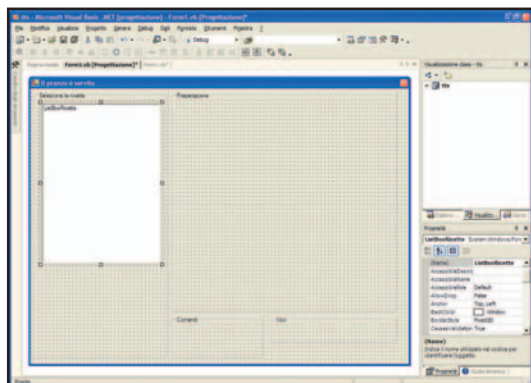
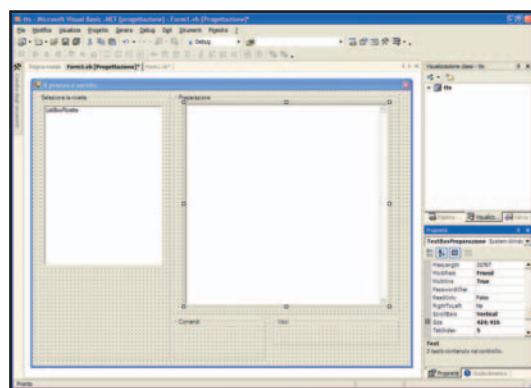
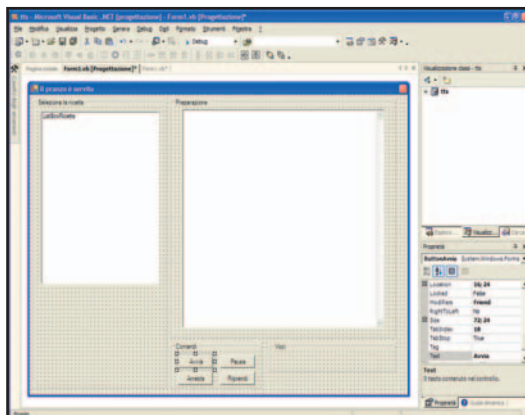
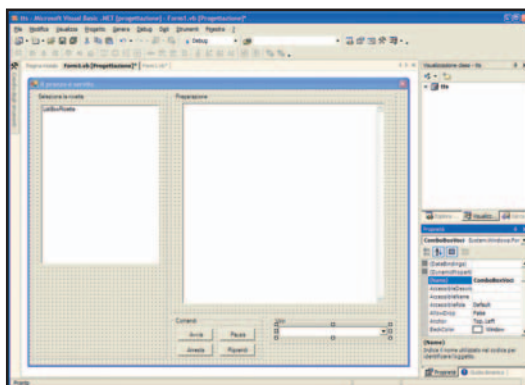
Questa fase di analisi consiste nella trasformazione del testo di ingresso, in cui ad ogni parola è stato già assegnato lo stress, nella corrispondente trascrizione fonetica, ovvero nella corrispondente sequenza dei suoni di base (fonemi) che appartengono al linguaggio. Il linguaggio italiano comprende 31 fonemi, di cui 22 sono consonanti, 7 vocali e 2 semivocali.
Il problema fondamentale di questa conversione è quello della risoluzione delle ambiguità, ovvero dei casi in cui ad un simbolo del linguaggio possono corrispondere più trascrizioni fonetiche.

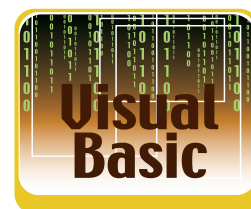
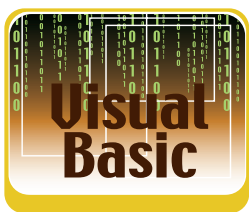
ANALISI PROSODICA

La componente prosodica di un sistema TTS, si occupa della analisi dei fenomeni riguardanti il rapporto e le variazioni di intensità (ampiezza), altezza (frequenza) e durata dei fonemi che costituiscono l'intera frase con l'obiettivo di fornire la pronuncia della frase stessa che ne determini il corretto significato. Essa rappresenta sicuramente la parte più complessa di un sistema TTS, in quanto coinvolge conoscenze che non sono interamente deducibili dal testo scritto. Alcune conoscenze di tipo linguistico, ad esempio quelle basate sulla struttura sintattica della frase, l'utilizzazione dei simboli di punteggiatura per la determinazione delle pause e della loro durata, nonché una analisi di tipo contestuale possono venire in aiuto per migliorare la prosodia delle frasi. In ogni caso, si può sicuramente affermare che l'analisi prosodica rappresenta la parte più complessa del linguaggio, ed è quella componente dei sistemi TTS verso la quale sono maggiormente rivolti gli sforzi della ricerca.

UN ESEMPIO CONCRETO

1 Per iniziare, creiamo un nuovo progetto Windows Applications e sulla finestra Form1 disegniamo l'interfaccia utente. Piazziamo un po' di controlli contenitore GroupBox per abbellire la finestra

La classe Stream è la classe base astratta di tutti i flussi. Un flusso è un'astrazione di una sequenza di byte, ad esempio un file, una periferica di input/output, o un socket TCP/IP. La classe Stream, e le relative classi derivate, forniscono una visualizzazione generica di questi diversi tipi di input e output, senza che lo sviluppatore debba conoscere i dettagli specifici del sistema operativo e delle periferiche sottostanti.

2 Selezioniamo un controllo ListBox dalla casella degli strumenti, e disegniamolo sulla form. Dalla finestra delle proprietà cambiamo il nome in ListBoxRicette. In ListBoxRicette visualizzeremo i nomi dei file contenenti le ricette.

3 Selezioniamo un controllo TextBox dalla casella degli strumenti, e disegniamolo sulla form. Dalla finestra delle proprietà cambiamo il nome in TextBoxPreparazione, la proprietà Multiline in True e la proprietà ScrollBars in Vertical . In TextBoxPreparazione visualizzeremo gli ingredienti e la preparazione della ricetta. Gli ingredienti dell'applicazione sono quasi pronti. Sarà necessario adesso aggiungere qualche controllo che ci consenta di determinare il flusso di esecuzione del nostro software

4 Selezioniamo quattro controlli CommandButton dalla casella degli strumenti, e disegniamoli sulla form. Dalla finestra delle proprietà cambiamo i nomi in ButtonAvvia, ButtonArresta, ButtonPausa, ButtonRiprendi. I quattro CommandButton serviranno rispettivamente per: Avviare, terminare, mettere in pausa e riprendere la lettura della ricetta. Ovviamente è in questi controlli che scriveremo il codice di gestione che farà uso delle Microsoft Speech Api. Come avremo occasione di vedere si tratterà di un meccanismo abbastanza semplice, ma molto potente

5 Infine Selezioniamo un controllo ComboBox dalla casella degli strumenti, e disegniamolo sulla form. Dalla finestra delle proprietà cambiamo il nome in ComboBoxVoci. In ComboBoxVoci visualizzeremo i tipi di voce (speech engine) installati nel sistema.

VISUALIZZARE LA RICETTA

Nel numero precedente ci siamo occupati della gestione degli elementi del file system ed abbiamo visto come sia garantita, in VB .NET 2003, dal namespace System.IO. Il namespace System.IO contiene una libreria di classi che offrono proprietà, metodi ed eventi per creare, copiare, spostare ed eliminare file e directory.
Le nostre ricette saranno contenute in file testo memorizzati in una directory di riferimento. Per leggere un file di testo dobbiamo utilizzare anche lo spazio dei nomi System.Text, che contiene classi per gestire la codifica dei caratteri ASCII, Unicode, UTF-7 e UTF-8, e classi astratte per la conversione

di blocchi di caratteri da e in blocchi di byte. Come al solito, per evitare di scrivere ogni volta il nome completo dei namespace, scriviamo le seguenti istruzione Imports:

```
Imports System
Imports System.IO
Imports System.Text
```

La classe Directory, permette la creazione, modifica e cancellazione delle directory e fornisce, inoltre, il metodo GetFiles(percorso). Il metodo GetFiles permette di recuperare tutti i file contenuti in una directory, restituendoli in una matrice di stringhe
La classe Path espone campi e metodi che consentono di ottenere informazioni su percorsi di file e directory. La classe Path fornisce alcuni metodi che permettono di estrarre informazioni dal percorso di un file. Se, ad esempio, applichiamo i metodi che utilizzeremo in seguito, al file C:\Ricette\Pizza Margherita.txt otteniamo:

- GetExtension restituisce l'estensione del file, quindi .txt.
- GetFileNameWithoutExtension restituisce il nome del file senza estensione, quindi Pizza Margherita

Dichiariamo una variabile in cui memorizzare la directory che dovrà contenere tutte le nostre ricette:

```
Dim pathRicette As String = "c:\ricette\"
```

Per visualizzare tutti i file di tipo testo contenuti nella directory di riferimento, scriviamo il codice necessario in una procedura dal nome MostraListaDeiFile

```
Private Sub MostraListaDeiFile ()
End Sub
```

Definiamo la variabile che dovrà contenere la matrice di stringhe, dei file presenti nella directory di riferimento, e la variabile che dovrà contenere il file corrente da mostrare nel ListBox. Utilizziamo il metodo GetFiles, passando come argomento la directory, per popolare la matrice dei file.

```
        Dim MatStrFile() As String
        Dim StrFile As String
        MatStrFile = Directory.GetFiles(pathRicette)
```

Utilizziamo il metodo Clear per pulire la ListBox, e con un ciclo For Each .. Next cicliamo sulla matrice dei file ed aggiungiamo ogni elemento in ListBoxRicette. Utilizziamo i metodi della classe Path per mostrare solo i file testo e visualizzare solo il nome del file senza estensione

```
        ListBoxRicette.Items.Clear()
        For Each StrFile In MatStrFile
            If Path.GetExtension(StrFile) = ".txt"
                                                Then
                        ListBoxRicette.Items.Add(
                             Path.GetFileNameWith
                              ut Extension(StrFile))
            End If
        Next
```

La lista delle ricette, deve essere mostrata, nel momento in cui l'applicazione viene eseguita, pertanto dobbiamo chiamare la procedura MostraListaDeiFile nell'evento Load della finestra Form1

```
Private Sub Form1_Load(ByVal sender As
     System.Object, ByVal e As System.EventArgs)
                              Handles MyBase.Load
MostraListaDeiFile ()
End Sub
```

Per effetto del codice scritto finora, appena viene avviata l'applicazione, verranno mostrati tutti i file (di tipo testo) contenuti nella directory. A questo punto l'utente potrà selezionare la ricetta da visualizzare in dettaglio, semplicemente cliccando con il mouse nella lista di sinistra. Il codice necessario dovrà essere scritto nell'evento SelectedIndexChanged della ListBox.

```
Private Sub
ListBoxRicette_SelectedIndexChanged(By Val sender
    As Object, ByVal e As System.EventArgs) Handles
                ListBoxRicette.SelectedIndexChanged
        Leggifile(pathRicette &
              ListBoxRicette.SelectedItem() & ".txt")
End Sub
```

LETTURA DI UN FILE

Siamo pronti ora per scrivere il codice necessario a mostrare i dettagli della ricetta, contenuti nel file selezionato. Per le operazioni di lettura e scrittura, su un file aperto, si deve utilizzare il metodo Open della classe File. Il metodo Open apre un oggetto FileStream, nel percorso specificato e con le modalità di accesso passate come argomento. Esistono inoltre tre varianti del metodo Open che

restituiscono un oggetto FileStream, nel nostro caso utilizziamo OpenRead che apre un file esistente per la lettura.
Filestream espone un oggetto Stream per un file, che fornisce il supporto per operazioni di lettura e scrittura sincrone e asincrone. Per leggere un blocco di byte dal flusso e scrivere i dati in un determinato buffer si può utilizzare il metodo Read. Il metodo Read restituisce zero solo dopo il raggiungimento della fine del flusso, pertanto si può usare questa condizione di test in un ciclo per essere sicuri di leggere l'intero contenuto del file.
Scriviamo il codice necessario, nella procedura dal nome Leggifile che riceve il nome del file da visualizzare in TextBoxPreparazione.

```
Private Sub Leggifile(ByVal NomeFile As
                                    String)
End Sub
```

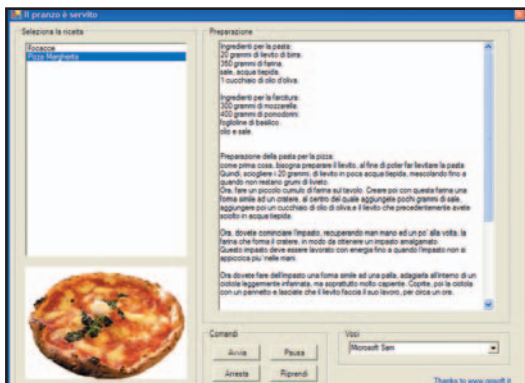



Fig. 2: *L'applicazione in esecuzione*

Per iniziare, definiamo la variabile oggetto di tipo FileStream ed apriamo il file selezionato

```
Dim fs As FileStream
fs = File.OpenRead(NomeFile)
```

Definiamo una variabile di tipo Byte, ed una variabile di tipo UTF8Encoding. La classe UTF8Encoding codifica i caratteri Unicode utilizzando UCS Transformation Format, a 8 bit (UTF-8). Supporta i valori di tutti i caratteri Unicode.

```
    Dim b(2048) As Byte
    Dim testo As UTF8Encoding =
                        New UTF8Encoding(True)
```

Infine leggiamo il flusso di dati, mostrandolo in TextBoxPreparazione, e chiudiamo l'oggetto di tipo FileStream

```
    Do While fs.Read(b, 0, b.Length) > 0
        TextBoxPreparazione.Text =
                        testo.GetString(b)
    Loop
    fs.Close()
```

IL METODO GETVOICES

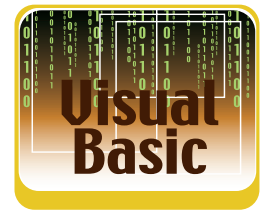
Possiamo tornare ad occuparci della sintesi e diamo la possibilità, se abbiamo più voci installate nel sistema, di selezionarne una, tra quelle disponibili, utilizzando il metodo GetVoices. Il metodo GetVoices restituisce la collezione delle voci installate accessibili tramite l'interfaccia ISpeechObjectToken, ed ammette un parametro opzionale che indica il criterio di selezione. Valorizzando il criterio di selezione, si può cercare uno SE, tra quelli installati, che risponda a determinate caratteristiche. In particolare i campi per cui si può selezionare uno SE sono:

- Language identifica il linguaggio dello SE
- Name Identifica il nome della voce
- Gender Identifica il sesso della voce
- Age Identifica l'età della voce
- Vendor Identifica il produttore della voce (es Microsoft)

Per utilizzare il criterio di selezione, dobbiamo usare la seguenti sintassi "Attributo = Valore". Possiamo scrivere ad esempio:

```
Voice.GetVoices("gender=female")
'per selezionare  le voci femminili
Voice.GetVoices("Gender=Male")
' per selezionare le voci maschili
```

In assenza del criterio di selezione, in GetVoices saranno contenuti tutti gli SE installati, ordinati alfabeticamente per nome. Se non ci sono voci che corrispondono al criterio, il metodo restituisce una collezione vuota. La configurazione di SAPI 5.1 contiene delle cartelle (folder) che rappresentano le risorse disponibili nel computer usate dalle SAPI: Il riconoscimento vocale (speech recognition, SR) e la sintesi vocale (text-to-speech, TTS). Queste cartelle sono organizzate in categorie come voci, lessici, e dispositivi audio di input.
L'oggetto SpObjectTokenCategory permette di accedere ad una categoria di risorse mentre SpObjectToken permette di accedere alla singola risorsa. Il metodo GetDescription, nel nostro caso, restituisce il nome del TTS selezionato Dichiariamo la variabile oggetto di tipo SpVoice

```
Dim Voice As SpVoice
```

E scriviamo il codice necessario nella procedura dal nome MostraElencoVoci

```
Private sub MostraElencoVoci()
End Sub
```

Naturalmente le voci dovranno essere caricate nel momento in cui si avvia l'applicazione, pertanto la chiamata alla procedura MostraElencoVoci dovrà essere contenuta nell'evento Load di Form1.
Definiamo la variabile di tipo IspeechObjectToken ed inizializziamo la variabile oggetto Voice di tipo SpVoice

```
        Dim Token As IspeechObjectToken
        Voice = New SpVoice
```

Cicliamo su tutte le voci disponibili e le inseriamo in ComboBoxVoci

```
        For Each Token In Voice.GetVoices
                        ComboBoxVoci.Items.Add
                        (Token.GetDescription())
        Next
        ComboBoxVoci.SelectedIndex = 0
```

Per leggere la ricetta con una voce diversa da quella di default, l'utente dovrà selezionare una voce tra quelle visualizzate nel ComboBox. Il codice necessario ad istanziare la voce selezionata, dovrà quindi essere scritto nell'evento SelectedIndexChanged del ComboBox.

```
Private Sub
                ComboBoxVoci_SelectedIndexChanged
                                    (ByVal sender
  As System.Object, ByVal e As System.EventArgs)
    Handles ComboBoxVoci.SelectedIndexChanged
        Voice.Voice = Voice.GetVoices().
                    Item(ComboBoxVoci.SelectedIndex)
    End Sub
```

IL PRANZO È SERVITO

A questo punto, non ci rimane che scrivere il codice associato ai quattro CommandButton, che ci permetteranno di ascoltare la ricetta mentre siamo ai fornelli.
Nell'evento click di ButtonAvvia scriviamo il codice necessario alla lettura della ricetta così come abbiamo visto nella parte iniziale dell'articolo, racchiudendolo in un gestore di errori Try..Catch

```
Private Sub ButtonAvvia_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
                        Handles ButtonAvvia.Click
        Try
            If Not TextBoxPreparazione.Text = ""
                                            Then
                Voice.Speak(TextBoxPreparazione.Text,
               SpeechVoiceSpeakFlags.SVSFlagsAsync)
            End If
        Catch ex As Exception
            MessageBox.Show("Errore di lettura",
                                        vbOKOnly)
        End Try
    End Sub
```

Nell'evento click di ButtonArresta scriviamo il codice necessario per concludere la lettura della ricetta. Per forzare la fine del processo di sintesi, dobbiamo utilizzare sempre il metodo Speak passando come argomenti: una stringa di valore Null ed il flag con valore SVSFPurgeBeforeSpeak per segnalare a SAPI che i dati rimanenti per essere sintetizzati devono essere scartati.

```
    Private Sub ButtonArresta_Click(ByVal sender
 As System.Object, ByVal e As System.EventArgs)
                        Handles ButtonArresta.Click
        Voice.Speak(vbNullString,
    SpeechVoiceSpeakFlags.SVSFPurgeBeforeSpeak)
    End Sub
```

Scriviamo infine il codice necessario per mettere in pausa il processo di sintesi e per riprendere dal punto in cui è stato interrotto. Allo scopo possiamo utilizzare i metodi Pause e Resume. Il metodo Pause mette in pausa il processo di sintesi nel punto più vicino e permette l'uso di altre voci. Il metodo Resume riprende il processo di sintesi dal punto in cui era stato messo in pausa

```
    Private Sub ButtonPausa_Click(ByVal sender As
      System.Object, ByVal e As System.EventArgs)
                        Handles ButtonPausa.Click
        Voice.Pause()
    End Sub
    Private Sub ButtonRiprendi_Click(ByVal sender
 As System.Object, ByVal e As System.EventArgs)
                        Handles ButtonRiprendi.Click
        Voice.Resume()
    End Sub
```

La nostra applicazione è terminata. L'uso delle Microsoft Speech API non è complesso. L'esempio che vi abbiamo proposto può facilmente essere adottato come template per la creazione di progetti successivi

Luigi Buono

SVILUPPARE CON VELOCITY

VEDIAMO COME SEPARARE LA PARTE PROGRAMMATIVA DA QUELLA PIÙ PROPRIAMENTE GRAFICA DI UN'APPLICAZIONE WEB, PER OTTENERE SOFTWARE FACILMENTE MANUTENIBILE. UNO DEI TOOL DI APACHE CI DARÀ UNA MANO

Il problema è sempre lo stesso. Per scrivere applicazioni dinamiche pronte per il web, il metodo classico prevede che in una pagina, scritta con un qualunque linguaggio, si mescolino: il codice necessario a gestire i dati, e i tag html necessari a mandare i dati in output. È un classico ad esempio che in una pagina dinamica ci sia la dichiarazione di una tabella e poi un ciclo di while che produce in output una serie di <tr><td>dato</tr></td> con dato che è ancora il frutto di una qualche elaborazione scritta con un linguaggio di alto livello. Questa non è sicuramente una soluzione ottimale, poiché tende a raddoppiare il lavoro di un programmatore, che non solo deve occuparsi di sviluppare il codice necessario alla gestione dei dati, ma anche di tradurre in modo opportuno e contestualizzare all'interno del codice il lavoro prodotto da un grafico. Ogni minima modifica grafica comporta che debba essere il programmatore a dover rivedere l'intero lavoro. In questo articolo tenteremo di separare il lavoro del programmatore da quello del grafico, utilizzando un tool sviluppato dal progetto Jakarta: Velocity. È un "engine" che si pone come "renderizzatore" di template. Il concetto è molto semplice. Ogni progetto si comporrà di almeno due file, uno contenente il codice, uno contenente il template. Nel codice verrà instanziato un oggetto di classe velocity e in qualche modo verrà stabilito qual è il file di template agganciato all'oggetto. Tutti i dati verranno poi manipolati in modo opportuno e "segnalati" al template tramite l'oggetto velocity. L'engine leggerà il template ed "espanderà" determinati tag al suo interno sostituendo ad essi i dati passati dal codice. In questo modo il programmatore dovrà solo occuparsi di trasferire al template i dati in un formato corretto, e il grafico di inserire nel template dei tag corrispondenti. I due lavori saranno separati. Vediamo come funzionerà il tutto.

COME FUNZIONA

Per cercare di comprendere la logica di funzionamento che sta alla base di Velocity, consideriamo un caso pratico: quello di una società che intenda sviluppare un'applicazione Web il cui obiettivo sia di permettere ai propri utenti di visualizzare le informazioni personali o relative ad un prodotto. In questo scenario, è necessario prima di ogni cosa che ciascun utente si sottoponga ad una procedura di login, effettuata la quale sarà possibile accedere alla pagina con i dati richiesti. Se intendiamo adoperare Velocity e rispettare il paradigma MVC, sappiamo bene che l'inserimento di codice Java nella pagina web non è il modo migliore per agire. Occorre invece offrire alla pagina Web una serie di elementi che, posizionati all'interno del codice HTML, generino il template l'output desiderato. Gli elementi di scripting consistono in statement logici e in istruzioni di accesso agli oggetti Java. Dal punto di vista prettamente sintattico, Velocity dispone di una collection chiamata Context, che si preoccupa di viaggiare attraverso i diversi livelli nel paradigma MVC. L'oggetto Context interagisce con i componenti controller e model e, così facendo, offre il template della pagina Web risultante. Si propone come un contenitore di dati che viene offerto al template ad esso collegato. Il codice che implementa Velocity effettua poi il parsing del template e sostituisce tutti gli elementi di scripting in esso contenuto con il testo ottenuto dagli oggetti nel contestocontext. Tutto questo, chiaramente, considerando una logica di base che prevede una certa coerenza nella scelta dei nomi da attribuire agli oggetti del contesto, che rimangono gli stessi nel passaggio da un livello all'altro. A ben vedere, una simile logica presuppone una forte interazione tra il progettista e lo sviluppatore, in una prospettiva d'insieme che porta ciascuno a fruire del lavoro dell'altro in perfetta interazione reciproca. Questo significa, quindi, che lo sviluppatore focalizza la propria attività sulla codifica dell'informazione necessaria all'interno del contesto: egli cioè integra la nuova pagina che intende realizzare all'interno del framework corrente, facendo attenzio-

ne a che al suo interno tutte le informazioni siano visualizzate ed estraendo i dati che gli servono dal progettista e posizionandoli all'interno del contesto. In altri termini:

```
Velocity.init();
VelocityContext contesto = new VelocityContext();
contesto.put( "nome", new String("Antonio Rossi") );
Template template =
        Velocity.getTemplate("account.vm");
StringWriter sw = new StringWriter();
template.merge(contesto, sw);
```

La porzione di codice vista adesso, che dovrebbe essere a cura dello sviluppatore, parte con l'inizializzazione dell'engine Velocity, operazione a cui segue quella legata alla creazione di un nuovo contesto. Utilizzando il metodo *put()*, un oggetto stringa viene assegnato alla chiave "nome" ed attaccato al contesto creato in precedenza. Successivamente, si farà riferimento al template creato dal progettista, che sarà richiamato opportunamente adoperando il metodo *getTemplate()*. Definito poi un oggetto StringWriter che gestisce l'output HTML, esso sarà utilizzato nell'operazione di merging che unirà il template al contesto.Il file account.vm che è un semplice file di testo a cura del grafico conterrà semplicemente

```
ciao $nome
```

quando il programma java verrà lanciato, si scoprirà che il suo output è gestito dal template account.vm. Il tag *$nome* contenuto nel file in questione verrà espanso sostituendo ad esso quello passatogli dal programma java. Il risultato dell'output sarà:

```
ciao Antonio Rossi
```

SVILUPPARE CON VELOCITY

Nell'esempio precedente, abbiamo visto come adoperare un semplice elemento di scripting insieme ad un contesto, allo scopo di produrre una pagina Web dinamica, nel rispetto del paradigma MVC. In realtà, Velocity offre numerose altre caratteristiche, che lo rendono un linguaggio estremamente versatile e potente. Occorre in primo luogo considerare le reference, distinte in variabili, proprietà e metodi. Per le prime, la cui notazione sintattica prevede la presenza del carattere $ seguito da un identificatore, il valore è generalmente determinato dal codice Java per mezzo del contesto. Le proprietà, identificate sintatticamente attraverso il carattere $ seguito da un identificatore, quindi un punto e poi un altro identificatore, si adoperano generalmente per ottenere l'attributo di un oggetto Java nel contesto, o per chiamare un metodo dell'oggetto (del tipo get) ed utilizzare il proprio valore di ritorno. I metodi invece sono chiamati all'interno del codice per far eseguire una serie di istruzioni che possono restituire un dato valore, e la sintassi è simile a quella delle proprietà. Vi sono poi le direttive, che consentono al progettista di controllare le reference. Gli elementi di scripting includono gli insiemi (assegnando ad una data variabile un dato valore), e altri elementi quali i costrutti di loop e quelli condizionali. Vi sono poi le Velocimacro, così chiamate perché si tratta di macro, contenenti codice HTML ed elementi di scripting, definite all'interno di Velocity. Velocity fa inoltre uso del sistema log4J per generare le istruzioni di logging all'interno di una data applicazione. Vi sono poi i resource loaders, che offrono il controllo sui template adoperati nella produzione di pagine Web, Anakia, un esempio di applicazione che consente all'XML di essere processato adoperando Velocity invece dell'XSL, e il supporto dei maggiori application servers sulla piazza come Resin, Tomcat e BEA WebLogic. Volendo iniziare a sviluppare con Velocity, è giusto prendere confidenza con alcuni statement del linguaggio, che saranno poi adoperati in maniera massiccia nello sviluppo di applicazioni future. Si può partire senza dubbio dallo statement #set, il cui utilizzo è riconducibile a quello fatto in un template per assegnare un valore ad un reference. Così, la porzione di codice:

```
#set( $nome = "Pippo" );
Ciao, $nome!
```

è alquanto semplice da comprendere: in essa viene definita la variabile nome e la si richiama all'interno di una stringa da far visualizzare in output. Se salviamo queste due righe all'interno di un file che chiameremo Ciao.vm, avremo realizzato il nostro primo template Velocity. Più tecnicamente, accade che il template, in aggiunta al parser Velocity, processerà in modo diretto il testo, mentre, incontrando la direttiva #set, crea una variabile interna al parser che gestisce la nuova variabile chiamata $nome, a cui viene assegnato il valore "Pippo". Vediamo ora di dedicarci a qualcosa di più complesso. Consideriamo la porzione di codice seguente:

```
import org.apache.velocity.app.Velocity;
import org.apache.velocity.VelocityContext;
import org.apache.velocity.Template;
import
    org.apache.velocity.exception.ParseErrorException;
```

IL PARADIGMA MVC

MVC è l'acronimo di Model View Controller. Si tratta di un metodo di programmazione che prevede una separazione netta del codice in tre strati. Il model si occupa di implementare le classi necessarie alla gestione dei dati e a tutti gli altri scopi previsti dal software. Il controller si occupa di gestire il flusso d'esecuzione del programma. La view infine si occupa di produrre l'output. I tre strati dell'applicazione devono essere sviluppati in file e codice separati, questo consente di poter "manutenere" il codice con un'efficacia notevole e consentire a tutti i membri di un'eventuale team di lavorare senza intralciare o essere intralciati dal lavoro degli altri membri.

```
import
 org.apache.velocity.exception.ResourceNotFoundExce
                                              ption;
import java.io.*;
import java.util.ArrayList;
public class Esempio {
public Esempio(String templateFile) {
try {
Velocity.init();
VelocityContext context = new VelocityContext();
Template template = null;
try {
template = Velocity.getTemplate(templateFile);
} catch(ResourceNotFoundException e2 ) {
System.out.println(
"Non posso trovare il template " + templateFile );
} catch(ParseErrorException e ) {
System.out.println("Errore sintattico nel template:
                                            " + e);
}
BufferedWriter writer = new BufferedWriter(
new OutputStreamWriter(System.out)
);
if ( template != null)
template.merge(context, writer);
writer.flush();
writer.close();
} catch( Exception e ) {
System.out.println(e);
}
}
public static void main(String[] args) {
Esempio t = new Esempio(args[0]);
}
}
```

In questo caso, il programma accetta come argomento di input un nome di file, che definisce il riferimento da adoperare come template. L'applicazione compie una serie di passi che diverranno abituali ad un utente esperto di Velocity: anzitutto inizializza l'engine Velocity, e ciò avviene attraverso il richiamo del metodo init(). Una volta inizializzato l'engine, è necessario caricare il template, e questo viene fatto utilizzando il metodo *getTemplate(String)*, dove l'argomento stringa è il nome del file passato. Da notare che la chiamata del metodo si inserisce in un ciclo di try/catch, dove l'eccezione è di tipo ResourceNotFoundException o *ParseErrorException*: il primo caso è legato ad un evento che si verifica se il file template specificato come parametro non può essere trovato, mentre il secondo gestisce errori di parsing verso l'oggetto template. Il passo successivo porta all'inizio del processo di generazione dell'output. Poiché si sta adoperando un'applicazione Java, il testo verrà prodotto dal template sulla console, anche se si potrebbe ricorrere ad altri sistemi di raccolta, come file, socket e così via. A questo punto, possiamo tracciare un quadro della situazione, con la presenza di numerosi oggetti attivi nella nostra applicazione: abbiamo infatti l'oggetto statico che implementa l'engine Velocity, l'oggetto Context, un oggetto Template creato al momento della lettura del template dal drive locale, e un oggetto BufferedWriter, che sarà adoperato per dirottare l'output verso la console attraverso lo stream System.out. Tornando al codice della nostra applicazione, si effettua un controllo sull'oggetto BufferedWriter, in modo che esso venga istanziato correttamente. L'applicazione del metodo merge() impone la presenza di due parametri: da un lato un oggetto di tipo Context e dall'altro un oggetto *BufferedWriter*. L'esecuzione del metodo comporta la fusione tra il contesto ed il template associato con i relativi elementi di scripting, in modo da creare l'output desiderato. Al termine, lo stream di output viene sottoposto a flush e chiuso. L'output generato dal template è la scrittura della frase di saluto con la variabile nome opportunamente valorizzata, per come atteso. Abbiamo visto come far riferimento alla direttiva #set per assegnare un valore al reference $nome. Tuttavia, non siamo ancora in grado di assicurare un comportamento dinamico alle nostre applicazioni. Occorre, in altre parole, riferirsi all'uso dell'oggetto Context dando la possibilità che questo sia fuso con il template. Proviamo perciò ad eliminare dal file Ciao.vm la prima riga, quella che introduce la direttiva *#set*. Andiamo poi nel file Esempio.java e, dopo la dichiarazione che inizializza l'oggetto context, inseriamo la linea seguente:

```
context.put("nome", "Pluto");
```

Salviamo quindi i due file, compiliamo e lanciamo nuovamente l'applicazione: il nuovo output sarà

```
Ciao Pluto!
```

Abbiamo perciò trasformato l'oggetto context in qualcosa di attivo e dinamico, grazie al metodo *put()* applicato ad esso.

IL RAPPORTO CON IL WEB

Dopo aver visto i comportamenti di Velocity in un ambiente standalone, è arrivato il momento di esaminare la situazione in ambito Web. Velocity, infatti, rappresenta una validissima alternativa al JSP, per lo sviluppo di applicazioni dinamiche e lineari, che abbiano il browsing quale elemento di applicazione. Consideriamo perciò il file seguente, che rappresenta il nostro template:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<elenco>
#foreach( $valore in $elenco )
```

```
<numero>$valore</numero>
#end
</elenco>
```

Si tratta di un file XML al cui interno è introdotta una nuova direttiva, *#foreach*. Essa consente di generare un ciclo basato su una o più reference: nel nostro caso, la reference è $elenco e c'è un oggetto vettore alimentato nel contesto. Il codice itera attraverso il vettore e visualizza così il valore associato *$valore*. Di seguito riportiamo il codice servlet necessario a produrre l'XML desiderato in output. Da notare che nel codice è creato un oggetto Vector, e ad esso sono aggiunti tre diversi valori. L'intero oggetto Vector è attaccato al contesto attraverso l'istruzione context.put("elenco", v).

```
import java.util.Vector;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import org.apache.velocity.Template;
import org.apache.velocity.context.Context;
import org.apache.velocity.servlet.VelocityServlet;
import
        org.apache.velocity.exception.Resource
                          NotFoundException;
import
  org.apache.velocity.exception.ParseErrorException;
public class VelocityServletExample extends
                             VelocityServlet {
public Template handleRequest( HttpServletRequest
                                     request,
HttpServletResponse response,
Context context ) {
Vector v = new Vector();
v.add("uno");
v.add("due");
v.add("tre");
context.put("elenco", v);
Template template = null;
try {
template = getTemplate("displaylist.vm");
} catch( Exception e ) {
System.out.println("Errore " + e);
}
return template;
}
}
```

Una volta che il vettore è posizionato all'interno dell'oggetto Context, il template adoperato per la visualizzazione dell'informazione del contesto è ottenuto dal server locale. Da notare che, invece di adoperare il metodo *merge()* dell'oggetto Template, si è scelto più semplicemente di ritornare l'intero oggetto template. Questa operazione, in realtà, non è implementata in modo diretto, nel senso che il template non è restituito direttamente all'utente: c'è invece un processo di background che effettua il merge tra il template "ritornato" ed il contesto. In output, è possibile visualizzare il file XML con i tre valori aggiunti all'entità *<numero>*.

UTILIZZARE TAG PERSONALIZZATI

Sviluppare per il Web utilizzando i tag JSP può apparire attività noiosa se rimane fine a se stessa. Grazie a Velocity, invece, può rappresentare un modo per realizzare applicazioni redatte ottimizzando i tempi di creazione e migliorando le prestazioni complessive. L'uso dei tag è alquanto immediato e comodo, così come è comodo utilizzare Velocity come engine per definire attraverso un template il cammino degli oggetti che vogliamo facciano parte della nostra applicazione. Tuttavia, come sappiamo, Velocity non può essere adoperato da solo per creare pagine che siano direttamente visibili attraverso un'interfaccia Web quale può essere un browser. Ecco allora che la soluzione da intraprendere diventa quella di fattorizzare il codice HTML in un file template, includere il codice che definisce il nostro tag all'interno di una normale classe Java, di riservare il descrittore del tag stesso ad un classico file XML, e di creare un file JSP che raccolga le informazioni e generi la pagina da prospettare sul browser. Un vantaggio ulteriore che deriva da questo approccio è quello legato al fatto che chi crea la libreria tag può testarne la generazione fuori dall'application server: pensiamoci, è estremamente comodo! Tornando agli elementi della nostra applicazione, occorre partire da una struttura dati ben definita. Supponiamo in particolare che l'applicazione implementi una normale videoteca e utilizzi una struttura database per il mantenimento delle informazioni relative ai titoli caricati. Possiamo pensare di adottare l'ambiente MySQL per l'implementazione di comandi che generino la tabella Film destinata ad ospitare i titoli in forza alla nostra videoteca. L'istruzione in codice sarà allora del tipo:

```
create table film (
id int not null primary key auto_increment,
titolo varchar(128),
regia varchar(64),
anno int);
```

Conoscendo la logica che sta alla base di Velocity, la classe Java è assai semplice da definire. Anzitutto, essa deve contenere al suo interno gli

NOTA

INSTALLARE VELOCITY

L'utilizzo di Velocity richiede l'installazione di Java 2 Virtual Machine. Se si vuole compilare Veloicity dal proprio codice sorgente, allora è richiesto J2SDK. Una volta scaricato il package che fa riferimento all'ultima release di Velocity, è necessario espanderlo all'interno di una cartella. È possibile poi effettuare il build dell'engine attraverso il tool Ant (reperibile dal sito http://ant.apache.org/) in una versione 1.3 o superiore. Dalla cartella in cui è statp estratto Velocity, al prompt dei comandi passare alla sottocartella build, quindi digitare l'istruzione ant, che provvederà a creare una directory bin all'interno della cartella di distribuzione di Velocity. La directory bin conterrà i file di classe compilati, un file Jar del tipo velocity-XX.jar (dove XX indica la versione corrente dell'engine). Prima di procedere, però, è necessario verificare che il classpath sia stato aggiornato in modo da includere il file Jar legato a Velocity.

elementi di validazione dei parametri tag, proprio nel punto in cui prende vita la logica di business personalizzata. Una volta soddisfatta la volontà di creare parametri tag e dopo averne testato la validità, occorrerà inserirli nell'oggetto Velocity Context che li renda accessibili al template di markup. Infine, faremo in modo che il tag engine configuri e generi il tag stesso verso l'output stream della pagina JSP.

```
VelocityContext myVelocityContext =
                                    getVelocityContext();
myVelocityContext.put("titolo", titolo);
myVelocityContext.put("regia", regia);
myVelocityContext.put("anno", anno);
myVelocityContext.put("film", film);
StringWriter myStringWriter = new StringWriter();
Velocity.mergeTemplate(myResource,
org.apache.velocity.runtime.RuntimeSingleton.
                    getString(Velocity.INPUT_ENCODING,
   Velocity.ENCODING_DEFAULT), myVelocityContext,
                                       myStringWriter);
pageContext.getOut().print(myStringWriter);
return EVAL_PAGE;
```

Il file di controllo taglib è un file XML che permette all'engine JSP di sapere quali tag stanno per essere pubblicati. Esso dichiara il nome del tag, la classe Java ad esso associata, e se un parametro è richiesto o meno:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE taglib PUBLIC "-//Sun Microsystems,
                    Inc.//DTD JSP Tag Library 1.1//EN"
   "http://java.sun.com/j2ee/dtds/web-
                              jsptaglibrary_1_1.dtd">
<taglib>
   <tlibversion>1.0</tlibversion>
   <jspversion>1.1</jspversion>
   <shortname>Cinema Tag Library</shortname>
   <uri>none</uri>
   <info>
      Definizione tag da usare per l'applicazione
                                          Videoteca.
   </info>
   <tag>
      <name>cinema</name>
      <tagclass>cinema.SampleTag</tagclass>
      <bodycontent>JSP</bodycontent>
      <info>
         Definizione tag cinema
      </info>
      <attribute>
         <name>film</name>
         <required>false</required>
         <rtexprvalue>true</rtexprvalue>
      </attribute>
   </tag>
</taglib>
```

Il file markup è configurato a nostro piacimento. È posto nel classpath e caricato opportunamente, e consente liberamente l'uso delle macro Velocity all'interno dello stesso file. Da parte nostra, abbiamo pieno accesso a tutti gli oggetti posizionati all'interno dell'oggetto Velocity Context. In qualunque momento, potremo creare anche macro più avanzate all'interno dello stesso file.

```
#macro (condAttribute $name $value)
#if ( $value )
#if ( $name )
${name}="${value}"#end#end#end
#condAttribute("Regia" $regia)
<table border="0" cellspacing="2" cellpadding="2"
                                     title="$titolo">
<tr>
   <td><b>Titolo</b></td>
   <td><b>Regia</b></td>
   <td><b>Anno</b></td>
</tr>
#foreach ( $id in $film )
<!—riga numero: ${velocityCount} -->
<tr>
   <td>$id.titolo</td>
   <td>$id.regia</td>
   <td>$id.anno</td>
</tr>
#end
</table>
```

Affinché sia possibile utilizzare il tag creato all'interno della pagina JSP, è necessario richiamarlo nella stessa pagina. Prima, però, è necessario definire una variabile che, opportunamente valorizzata, rappresenterà poi l'argomento da passare al tag in questione. Scriveremo perciò:

```
<%
         java.util.ArrayList film = null;
         cinema.CinemaTag st = new
                                  cinema.CinemaTag();
         film = st.getElencoFilm();
%>
```

avendo avuto cura di implementare opportunamente, all'interno della classe CinemaTag, il metodo *getElencoFilm()*:

```
public ArrayList getElencoFilm()
   {
         Map titoli = new HashMap();
      ArrayList film = new ArrayList();
      Connection con=null;
         try{
```

```
            //Istanzio il Driver
  Class.forName("org.gjt.mm.mysql.Driver")
          //inizializzo l'oggetto con
con=DriverManager.getConnection("jdbc:mysql://
                              localhost:3306/film");
            // Istanzio e inizializzo l'oggetto st
                                 di tipo Statement
            Statement sta = con.createStatement();
            //Istanzio e inizializzo l'oggetto rs di tipo
                                         ResultSet
                  String query = "SELECT * FROM
                            Film ORDER BY Titolo";
                  ResultSet rs =
                               sta.executeQuery(query);
                  while(rs.next()){
                  Map map = new HashMap();
                  int id = rs.getInt(1);
                  String titolo = rs.getString(2);
                  String regia = rs.getString(3);
                  int anno = rs.getInt(4);
                  map.put("id", id);
                     map.put("titolo", titolo);
                     map.put("regia", regia);
                     map.put("anno", anno);
                     titoli.put("titoli", titolo);
                     map.put("titoli", titoli);
                             film.add(map);
                             String output =
                          doStandaloneTest(map);
                     System.out.println(output);
                  }
                  setFilm(film);
                  sta.close();
                  con.close();
          } catch (ClassNotFoundException e){
          System.out.println("Impossibile caricare il
                                   driver: "+ e);
                  } catch (SQLException e){
          System.out.println("Errore SQL: "+ e);
          } catch (Exception e) {
          // TODO Auto-generated catch block
                  e.printStackTrace();
          }
          return film;
    }
```

La chiamata del tag in questione sarà quindi implementata attraverso l'istruzione seguente:

```
<cin:cinema film="<%=film%>"/>
```

In maniera analoga, potremo implementare la ricerca dei titoli all'interno del catalogo, e definire una sorta di menu da far comparire in home page. Ancora, la pagina JSP che implementa la ricerca dovrà basarsi su un metodo opportunamente definito all'interno della classe CinemaTag:

```
<html>
<head>
<title>
Videoteca
</title>
</head>
<link rel="stylesheet" type="text/css"
                                    href="layout.css">
<body>
<div class="header">
          <H1><span class="button"><center>Vi
         deoteca EMOTIONS</center></span></H1>
          <p>
<%

%>
          <form method="post"
                          action="CinemaRicerca.jsp">
<%
          String titolo = request.getParameter("titolo");
          java.util.ArrayList film = null;
          System.out.println("titolo: "+titolo);
          if (titolo==null || titolo.equals("")) {
%>
                    Cerca per titolo: <input
         type="text" name="titolo" maxlength="20">
                    <input type="submit"
                         name="Cerca" value="Vai" >
<%
          } else {
                    cinema.CinemaTag st = new
                                  cinema.CinemaTag();
                    film = st.cercaFilm(titolo);
                    if (film.size()==0) {
%>
                    Siamo spiacenti, ma non ci sono
                         film che soddisfano la ricerca!
                    <p/>
                    <a href="Prima.jsp"><span
       class="navAlpha">Torna indietro</span></a>
<%
                    } else {
%>
                    <cin:cinema
                                 film="<%=film%>"/>
                    <p/>
                    <a href="Prima.jsp"><span
       class="navAlpha">Torna indietro</span></a>
<%
                    }
          }
%>
          </form>
</div>
</body>
</html>
```

Luigi Caputo

CREARE MACRO CON VELOCITY

Tra le direttive in forza a Velocity, quella #macro, di cui offriamo un accenno anche nell'esempio che chiude l'articolo, offre un meccanismo per il riuso del codice template. Piuttosto che importare e processare tutto il codice template contenuto all'interno di un file arbitrario, la direttiva #macro offre la sintassi per specificare e identificare un blocco di codice template, comprensivo dei parametri di input di supporto. Il blocco può essere specificato sia in una libreria macro che in un classico file template.

UN MOTORE DI RICERCA CON AJAX

AJAX È SICURAMENTE LA NOVITÀ DEL MOMENTO. STA CONTRIBUENDO ALLA RIVOLUZIONE DEL WEB GRAZIE AL PROLIFERARE DI APPLICAZIONI CHE TRAGGONO ENORME VANTAGGIO DAL SUO UTILIZZO, VEDIAMO COSA È E COME FUNZIONA.

L'idea è semplice: in un'applicazione web dinamica gli elementi che realmente cambiano nel passare da un a pagina all'altra sono veramente pochi. Ad esempio la pagina successiva a quella di login manterrà pressapoco la stessa grafica della pagina precedente, e solo il box con le informazioni sull'autenticazione avrà un aspetto differente. Allora perchè ricaricaricare l'intera pagina? posso fare in modo che solo il box che contiene il login vari, lasciando il resto invariato. Una sorta di iFrame molto avanzato, perchè inglobato nella struttura di una stessa pagina. Vediamo come funziona. Quando per la prima volta ho utilizzato il servizio di Google Suggest, ancora in versione beta, sono rimasto a dir poco esterrefatto: ma come fanno a proporre la lista dei risultati in tempo reale? E con tempi di riposta così veloci? Ovviamente hi curiosat nel codice javascript e qui ho notato l'uso dell'oggetto XmlHttpRequest. Incuriosito dal corretto funzionamento sul browser firefox, sono partito alla ricerca delle specifiche sulle funzionalità e qui ho notato la somiglianza con l'ActiveX XmlHttpRequest di Microsoft, compatibile solo con internet explorer e sviluppato per l'utilizzo in Outlook Web Access da ormai cinque anni. L'XmlHttpRequest è, invece, attualmente implementato da diversi browser: Firefox, Safari, Konqueror ed Opera in tutte le sue versioni, inclusa quella per SymbianOS. La sua diffusione ha contribuito a creare un standard de facto che ha successivamente preso il nome di Ajax (Asyncronous Javascript and Xml).

QUANDO E PERCHÉ USARE AJAX

La larga diffusione di javascript e dell'XmlHttpRequest non è sicuramente il motivo principale che deve spingerci ad usare la metodologia Ajax. Il principio alla base è legato alla possibilità di evitare inutili refresh e conseguenti carichi di richieste di tutta la pagina quando la porzione da aggiornare è limitata ad un singolo elemento o controllo, come una combobox. Il classico esempio di pratica deleteria può essere l'aggiornamento del box delle news di un portale. Prevedendo un aggiornamento delle news ad intervalli di tempo troppo stretti tra loro, si rischia di generare una quantità troppo alta di richieste, ottenendo l'esatto effetto contrario. L'intento di Ajax è quello di produrre interfacce client più potenti e ricche di funzionalità, più vicine alle esigenze degli utenti. Non bisogna estremizzare questo concetto, però. L'uso va attentamente valutato in ogni applicazione web, in ogni singola pagina. Come molti di voi sapranno, javascript non è supportato, in tutto o in parte, su diversi browser, ed un uso intensivo di javascript può minare l'usabilità del sito se non sono state previste le adeguate funzionalità sostitutive.

Ho ritenuto necessaria questa breve premessa per meglio capire ora cosa, invece, Ajax può fare per noi passando all'analisi di un applicazione di esempio. Nell'articolo cercheremo di costruire un piccolo motore di ricerca di documenti contenuti in una directory del nostro sito ed utilizzando la libreria Ajax.NET Professional sviluppata da Michael Schwarz recupereremo i dati senza la necessità di effettuare aggiornamenti della pagina.

L'OPEN SOURCE DI AJAX.NET PROFESSIONAL

Innanzitutto scarichiamo il componente Ajax.NET Professional, che d'ora in poi chiameremo per comodità Ajax.NET, seguendo il link presente nel box laterale o preleviamolo dal cd allegato alla rivista copiandolo in una cartella di riferimento sul nostro pc, ad esempio c:\Assembly\Ajax. Creiamo un nuovo progetto ASP.NET ed aggiungiamo un riferimento all'assembly ajax.dll appena copiato. Il componente funziona sfruttando le caratteristiche degli HttpHandlers. Un HttpHandler è un filtro in grado di gestire tutte le richieste che pervengono verso una determinata risorsa ed Ajax.NET è com-

pletamente basato su questa feature. E' necessario, infatti, dichiarare nel file di configurazione web.config la seguente stringa:

```
<configuration>
  <system.web>
    ...
    <httpHandlers>
        <add verb="POST,GET"
            path="ajaxpro/*.ashx" type="AjaxPro.
                AjaxHandlerFactory, AjaxPro" />
    </httpHandlers>
    ...
  </system.web>
</configuration>
```

Fig. 1: ***Il form che andremo a costruire.***

Aggiungiamo al progetto la pagina di default ed inseriamo una textbox searchBox ed un button searchBtn. Il form è simile a quello visualizzato in Per consentire al framework Ajax.NET di generare gli script lato client per le funzioni server è necessario registrare la classe contenitore. Il namespace Ajax ha una classe Utility con diversi metodi statici tra cui RegisterTypeForAjax, che si occupa di registrare, per la generazione degli script sul client, il tipo che lato server risolve ed evade le richieste. Nell'evento Load della pagina inseriamo la registrazione del tipo della pagina, contenitore dei servizi ajax:

```
private void Page_Load
        (object sender, System.EventArgs e)
{
AjaxPro.Utility.RegisterTypeForAjax
                (typeof(AjaxSearchPage));
}
```

E' doveroso precisare che la classe non necessariamente deve corrispondere alla pagina in esecuzione, come avviene per semplice comodità nel nostro articolo. Ora possiamo procedere con la creazione del metodo, o dei metodi, che costituiranno i nostri "servizi ajax". Se avete scritto Web Service con .NET l'approccio vi sembrerà molto familiare perché anche Ajax.NET fa uso degli attributi, per dichiarare un metodo come servizio ajax, anteponendo l'attributo AjaxMethod, contenuto nel namespace Ajax, al metodo stesso. La nostra funzione di ricerca avrà quindi la seguente sintassi:

```
[AjaxPro.AjaxMethod()]
public DataTable GetFileList(string filter)
```

E' ora possibile utilizzare da semplici script javascript, le funzioni esposte lato server, al pari di funzioni javascript client. Ajax.NET, infatti, espone sul client i servizi ajax con la stessa firma dichiarata lato server, con in più la possibilità di utilizzare ogni singolo metodo sia in modalità sincrona che in modalità asincrona. Infatti, la funzione accetta un ulteriore parametro che definisce una funzione di callback da richiamare per l'elaborazione della risposta al termine dell'esecuzione. Se il parametro viene specificato, la funzione sarà eseguita in modalità asincrona, altrimenti sarà eseguita in modalità sincrona. Nel nostro esempio utilizzeremo la modalità asincrona, come riportato nel seguente codice:

```
<script language="javascript">
function DoSearch()
{
    var searchBox =
        document.getElementById("searchBox");
ioProgrammo.Ajax.AjaxSearchPage.GetFileList
        (searchBox.value, DoSearch_Callback);
}
function DoSearch_Callback(res)
{
    if(typeof(res) == 'object')
    {
        var dt = res.value;
        var searchResult =
    document.getElementById("searchResult");
        var html = "<table>";
html += "<tr>";
html +="<td> </td>";
html += "<td><b>Nome</b></td>";
html += "<td><b>Tipo</b></td>";
html += "<td><b>Dimensioni</b></td>";
html += "<td><b>Data
            di Creazione</b></td>";
html += "</tr>";
for(var i=0;
            i<dt.Rows.length; i++)
        {
html += "<tr>";
html += "<td><img src='images/"
        + dt.Rows[i].Type + ".gif'></td>";
html += "<td>" + dt.Rows[i].
                Name + "</td>";
html += "<td>" + dt.Rows[i].Type + "</td>";
html += "<td>" + dt.Rows[i].Size + "</td>";
html += "<td>" + dt.Rows[i].
```

ESTENDERE I TIPI DI AJAX.NET PROFESSIONAL

Ajax.NET Professional è in grado di gestire, nella proprietà Value, diversi tipi nativi di .NET tra cui string, integer, double, boolean, ma la caratteristica più importante è la conversione di oggetti complessi come DataSet, DataTable, DataRow, DataRowView, DataView, DateTime e gli arrays attraverso le interfacce IDictionary, IEnumerable, IList e molti altri. E' possibile estendere le classi converter predefinite sviluppando le proprie semplicemente implementando l'interfaccia IJavaScriptConverter.

DOWNLOADS

La libreria Ajax.NET Professional sviluppata da Michael Schwarz è liberamente scaricabile dal sito http://www.schwarz-interactive.de dove è possibile trovare ulteriori esempi. Il framework Atlas, nella sua versione beta di dicembre 2005 sulla quale è basata il presente articolo, è scaricabile dal seguente indirizzo: http://msdn.microsoft.com/asp.net/info/future/atlastemplate

COME NASCE AJAX

La definizione di AJAX (Asyncronous Javascript and Xml) si deve ad un articolo di Jesse James Garret che nel febbraio del 2005 la definisce non come una tecnologia, ma come un insieme di diverse tecnologie. In Ajax coesistono infatti XHTML, CSS, DOM, XML, XSLT, XMLHttpRequest ed infine JavaScript che le collega tutte. Il testo completo dell'articolo è reperibile al seguente indirizzo: http://adaptivepath.com/publications/essays/archives/000385.php

```
                    CreationTime + "</td>";
                        html += "</tr>";
                    }
searchResult.innerHTML = html + "</table>";
            }
    }
    </script>
```

come è possibile notare, nella funzione DoSearch viene chiamato il metodo AjaxSearchPage.GetFileList, completo di namespace così come dichiarato sul server, passando come parametro il filtro di ricerca e la funzione di callback. In DoSearch_Callback, la funzione di callback, il framework Ajax.NET passa un oggetto contenente quattro proprietà:

- **value**: Il valore di ritorno della funzione, equivalente, per esempio, ad una stringa, ad un dataset o anche ad un tipo personalizzato;
- **error**: In caso di eccezione è l'istanza di un oggetto con tre proprietà: name, contenente il tipo di eccezione, description, che riporta il contenuto della proprietà Message dell'eccezione generata sul server, e number contenente un identificativo univoco. L'utilizzo diretto della variabile nella forma res.error equivale a richiamare il metodo toString() che restituisce l'unione delle proprietà name e description;
- **extend**: Consente di creare una classe per la persistenza delle informazioni e l'eventuale passaggio di dati tra la funzione chiamante e la funzione di callback. Per approfondire questo argomento vi rimando al sito ufficiale il cui indirizzo è riportato sul box laterale;

Ajax.NET è in grado di gestire, nella proprietà Value, diversi tipi nativi di .NET tra cui string, integer, double, boolean, ma il punto di forza è la capacità di ricreare lato client gli oggetti DataSet, DataTable, DateTime e gli arrays. Per i tipi non direttamente gestiti, viene restituito il valore dato dal metodo ToString(). Sfruttando questa caratteristica, la funzione di callback DoSearch_Callback elabora la risposta leggendo la proprietà Value come un normale oggetto DataTable e sfruttando la collection Rows.

ATLAS: LA SOLUZIONE TARGATA MICROSOFT

Con il rilascio di ASP.NET 2.0 sono stati fatti passi da gigante nello sviluppo di applicazioni web, grazie all'introduzione di diverse novità oltre a migliorare e perfezionare quanto già esistente. Lo sviluppo della nuova release era, però, già ad uno stadio troppo avanzato e la Microsoft non poteva permettersi di

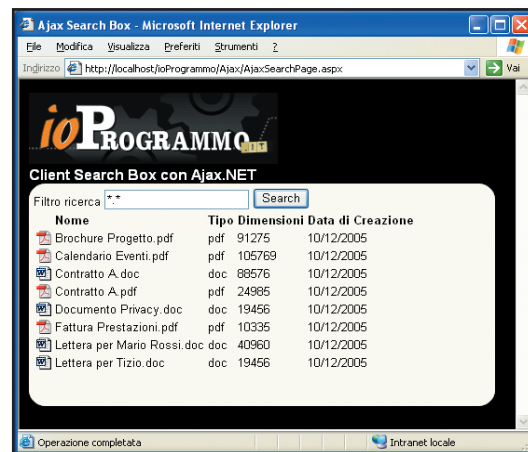



Fig. 2: risultato della ricerca fatta con AjaxPro.NET.

rimanere estranea all'evoluzione che Ajax stava via via portando allo sviluppo web. Perciò ecco che viene proposto Atlas. Ancora in versione beta, Atlas propone un set di controlli integrati con il framework ASP.NET che consentono la creazione di complesse funzioni Javascript sia in stile Ajax sia per la modifica dinamica delle web form. Atlas, infatti, supporta due tipologie di controlli:

- **ASP.NET 'Atlas' Client-side Controls**: consentono la creazione di script per l'iterazione diretta dei controlli con gli eventi generati dall'utente o da altri script della pagina. Il framework è così in grado di evitare continui refresh della pagina, oltre a semplificare l'utilizzo di caratteristiche complesse come il drag'n'drop. Si tratta, in realtà, di una sorta di evoluzione di dhtml;
- **ASP.NET 'Atlas' Server Controls**: è la reale implementazione di ajax in quanto permettono l'uso dei controlli client per lo scambio di informazioni con l'applicazione Web lato server. Attraverso degli extenders è possibile aggiungere funzionalità ai controlli ASP.NET esistenti. Un esempio: la funzionalità di autocomplete per il controllo textbox è appunto realizzata come control extender;

Per effettuare il porting del form di ricerca oggetto del nostro articolo in ASP.NET 2.0/Atlas, utilizzeremo esclusivamente gli Atlas Server Controls. Dopo aver scaricato ed installato il pacchetto indicato nel box laterale, apriamo Visual Studio 2005, creiamo un nuovo progetto Atlas, come raffigurato in figura 3, e creiamo una nuova pagina AtlasSearchPage.aspx. Aggiungiamo alla pagina un controllo asp:textbox ed un controllo asp:button ed impostiamo l'id rispettivamente come searchBox e searchBtn. Ora dobbiamo definire la porzione di pagina che deve essere aggiornata al verificarsi di uno o più eventi. Per questo scopo, Atlas mette a disposizione il controllo UpdatePanel che si occupa di definire gli eventi, chiamati triggers, e la modalità

di visualizzazione del set di controlli ad essi correlati. Nel designer di Visual Studio 2005, quindi, aggiungiamo un controllo Atlas:UpdatePanel dalla

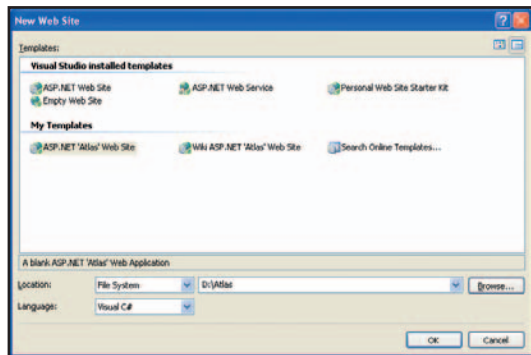



Fig. 3: **risultato della ricerca fatta con AjaxPro.NET.**

toolbox laterale e trasciniamo al suo interno un controllo GridView per la visualizzazione dei risultati. Nelle proprietà del controllo UpdatePanel clicchiamo sulla collection Triggers e successivamente clicchiamo su "New Trigger" per inserire la regola che genera l'aggiornamento del pannello, così come riportato in figura 5. Analizzando il sorgente della pagina dovremmo avere un risultato simile a questo:

```
<atlas:UpdatePanel ID="updPanel1" runat="server">
<Triggers>
<atlas:ControlEventTrigger ControlID=
                        "searchBtn" EventName="Click" />
</Triggers>
<ContentTemplate>
<asp:GridView ID="searchResult" GridLines=
                        "None" HeaderStyle-HorizontalAlign=
                        "Left" runat="server"></asp:GridView>
</ContentTemplate>
</atlas:UpdatePanel>
```

Per il corretto funzionamento dello script, però, manca ancora un piccolo tassello: un controllo che si occupa dell'orchestrazione delle funzionalità lato-client. Per questo Atlas si avvale del controllo ScriptManager, il vero cervello di una

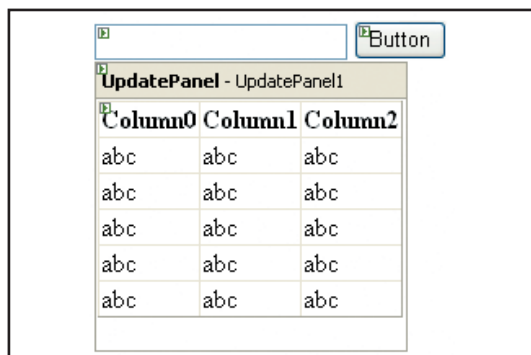



Fig. 4: **L'update panel visualizzato nel design di Visual Studio .NET 2003.**

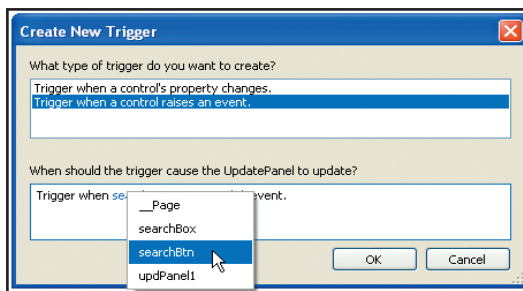



Fig. 5: **La selezione del controllo e dell'evento che generano l'aggiornamento del server.**

pagina ASP.NET-Atlas attraverso il quale possiamo, ad esempio, controllare il tipo di rendering o gestire i riferimenti agli script. Trasciniamo il controllo Atlas:ScriptManager dalla toolbox laterale ed impostiamo la proprietà EnablePartialRendering su true. Questa proprietà consente l'interazione con l'UpdatePanel attraverso il classico modello ad eventi di ASP.NET. E' ora infatti sufficiente creare il gesto-

Fig. 6: **La funzione di ricerca eseguita con Microsoft Atlas**

re di eventi lato server ed Atlas eseguirà il metodo effettuando l'aggiornamento solo della porzione di pagina definita nell'UpdatePanel. Sfruttando la stessa funzione di ricerca GetFileList utilizzata per l'esempio Ajax.NET otterremo un risultato simile a questo:

CONCLUSIONI

In questo articolo abbiamo visto come è possibile utilizzare Ajax con ASP.NET 1.x e anche come sarà possibile farlo con ASP.NET 2.0 attraverso l'uso del nuovo framework. In definitiva Ajax rappresenta un elemento di discontinuità rispetto alla programmazione Web tradizionale. a fronte di qualche riga di codice in più consente velocità e funzionalità

Fabio Cozzolino

Microsoft IXMLHTTPRequest
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/xmlsdk/html/xmobjxmlhttprequest.asp

Mozilla XMLHttpRequest
http://developer.mozilla.org/en/docs/XMLHttpRequest

Sito Ufficiale Ajax.NET Professional
http://www.schwarz-interactive.de

Il blog di Michale Schwarz
http://weblogs.asp.net/mschwarz

Sito Ufficiale Atlas
http://atlas.asp.net

Il mio blog
http://blogs.ugidotnet.org/fabioc

XML E GESTIONE DEI FILE CON PHP

IN QUESTO ARTICOLO VEDREMO COME RECUPERARE I DATI DA UN FILE XML, COME CREARE UN NOSTRO FILE XML E INFINE COME SALVARE TUTTO SU DISCO. TECNICHE SEMPRE UTILI SPECIALMENTE PER APPLICAZIONI DI INTESCAMBIO DATI.

Tutti conoscono ormai XML, si tratta di uno standard per la descrizione formale di qualunque tipo di dato. La definizione non è precisa, ma serve ai nostri scopi. Ad esempio il file redattore.xml potrebbe contenere qualcosa del genere

```
<redattore>
        <nome>
                Pippo
        </nome>
        <ruolo>
                Cronaca
        </ruolo>
</redattore>
<redattore>
        <nome>
                Paperino
        </nome>
        <ruolo>
                Sport
        </ruolo>
</redattore>
```

Si intuisce facilmente che questo file XML contiene due nodi redattore, ciascun redattore ha un nome e un ruolo.
Non ci dilungheremo in questo articolo sul formato XML, sul concetto di radice etc. Diamo per scontato che chi legge abbia le nozioni di base su XML sufficienti a utilizzare questo formato con PHP.
In ogni caso il nostro scopo è leggere il file, scansionarlo riga per riga e estrarre le informazioni che ci servono. Vedremo come questo sia possibile con le varie versioni di PHP.

LA CLASSE DOMDOCUMENT

Inizieremo con qualche esempio basato su PHP5 e DOM. Al momento non è importante conoscere le differenze fra i vari modelli, chi vuole può leggere il box di approfondimento in questo stesso articolo. E' invece importante sapere che con PHP4 è necessario utilizzare un modello basato su SAX mentre per PHP5 possiamo usare comodamente DOM.
Il punto di partenza di tutto è la classe DomDocument. Il costruttore semplicemente alloca lo spazio per contenere un documento di tipo XML. Il documento in questione può essere caricato dall'esterno e poi parserizzato, oppure è possibile il contrario, ovvero costruire un documento XML da esportare per altri fini.
Per caricare un documento XML dall'esterno è possibile utilizzare il metodo load.
Il metodo load è in grado di caricare ogni stream consentito dal linguaggio. Per cui è possibile sia caricare documenti XML locali sia documenti XML residenti in rete.
Considerate ad esempio:

```
<?
        $xmldocument=new DOMDocument();
    $xmldocument
  >load("http://www.theregister.co.uk/excerpts.rss");
?>
```

Questo estratto di codice non fa altro che creare un nuovo oggetto di tipo DOMDocument e riempirlo prelevando dalla rete un file XML, nel nostro caso in formato rss ovvero il celebre formato che consente di importare dati da un sito all'altro nella rete.
A questo punto l'oggetto $xmldocument contiene il file in questione in un formato tale da potere essere gestito in modo efficace dai metodi della classe.

Il file rss in questione ha il seguente formato

```
<rss version=20062.02006>
    <channel>
    <title>The Register</title>
    <link>http://www.theregister.co.uk/</link>
```

```
        <description>Biting the hand that feeds
                                    IT</description>
<copyright>Copyright 2005, Situation
                         Publishing</copyright>
        <image>
                <title>The Register</title>
                        <width>88</width>
                        <height>31</height>
                        <url>
                        http://www.theregister.co.
        uk/Design/graphics/Reg_default/The_Register_
                                        RSS.png
                        </url>
        <link>http://www.theregister.co.uk/</link>
        </image>
        <language>en-GB</language>
<webMaster>webmaster@theregister.co.uk
                                    </webMaster>
        <lastBuildDate>Fri, 21 Jan 2005 09:07:24
                              GMT</lastBuildDate>
        <ttl>120</ttl>
        <item>
                <title>Europe goes mad for
                                broadband</title>
                <link>
                        http://go.theregister.
          com/feed/2005/01/21/forrester_broadband/
                </link>
                <description>
                        <h4>Massive price
                                        cuts</h4>
                        <p>Almost half
                                        [...]</p>
                </description>
                <pubDate>Fri, 21 Jan 2005
                        09:06:49 GMT</pubDate>
        </item>
        <item>
                <title>
                        How Dell made North
                        Carolina beg for business
                </title>
                <link>
                        http://go.theregister.
                com/feed/2005/01/21/dell_nc_beg/
                </link>
                <description>
                    <h4>Memos reveal patriot
                                        games</h4>
                <p>Any reporter who has [...]</p>
                </description>
                <pubDate>Fri, 21 Jan 2005
                        01:46:38 GMT</pubDate>
     </item>
   </channel>
</rss>
```

Si tratta di un formato dati per presentare le notizie dal web, abbastanza diffuso. Il nostro problema è dunque muoverci nell'albero XML per recuperare di volta in volta i dati che ci servono per creare una pagina HTML che li rappresenti.

IL METODO GETELEMENTS BYTAGNAME

Ricerca all'interno di un documento XML tutti gli elementi relativi a un tag dato ritorna un oggetto di classe DomDomList che contiene i dati in questione. Il contenuto dell'oggetto in questione può essere recuperato attraverso un ciclo di foreach per esempio e stampato a video tramite il metodo textcontent.

Considerate ad esempio:

```
$xmldocument=new DOMDocument()
$xmldocument
   >load("http://www.theregister.co.uk/excerpts.rss");
$nodelist=$xmldocument
                    >getElementsByTagName("title");
foreach ($nodelist as $node) {
       print $node->textContent."<br>";
}
```

restituisce

```
The Register
```

PHP 4 E PHP 5

Nella versione 4 di PHP esisteva un supporto a XML sufficientemente interessante ma necessariamente limitato. XML, pure essendo uno standard, solo relativamente recentemente ha assunto una posizione di grande rilievo. Così mentre PHP4 prevedeva un supporto a XML basato su SAX, ovvero un sistema che consentiva un parsing efficace ma non semplicissimo da implementare, in PHP5 questo supporto è stato ampiamente migliorato.
In PHP5 tutto il supporto a XML è stato affidato alla libreria libxml2 sviluppata dallo gnome project. Questo tipo di approccio consente una interoperabilità stretta fra le varie tecniche ed estensioni che possono essere adottate nel parsing di un documento XML.
In particolare il modello SAX che basa il suo parsing su funzioni di CallBack è stato mantenuto e questo fa si che il vecchio codice sviluppato con PHP4 continui a funzionare, tuttavia mentre nelle versioni di PHP precedenti alla 5 , SAX si basava su una libreria chiamata expat, nelle nuove versioni si basa sulla libreria libxml2 di gnome, e questo potrebbe provocare qualche piccola incompatibilità se avete installato una libreria libxml2 non troppo recente.
Il modello DOM implementato in PHP5 segue fedelmente le indicazioni del W3C a differenza di quello implementato in PHP4 che invece le seguiva solo parzialmente. Se avete scritto codice PHP nella versione 4 facendo uso di DOM è molto probabile che il nuovo codice non funzioni in PHP5, questo perché i metodi e le proprietà aderiscono adesso formalmente agli standard. E' invece molto probabile che non avrete difficoltà a usare questo metodo se provenite da altri linguaggi che già ne fanno uso.

```
The Register
Europe goes mad for broadband
How Dell made North Carolina beg for business
```

Il che è corretto solo in parte. In effetti il metodo ha restituito tutti gli elementi 'title' senza tener conto della gerarchia. Mentre il nostro scopo era quello di mostrare i titoli delle notizie, ci viene restituito un elenco contente anche le informazioni relative al 'title' del sito che esporta le notizie e quello relativo al title del tag 'image'. A noi realmente interessavano solo i contenuti del tag 'title' del ramo 'items'.
Una versione corretta dell'algoritmo di cui sopra potrebbe essere la seguente:

```
$xmldocument=new DOMDocument();
$xmldocument
  >load("http://www.theregister.co.uk/excerpts.rss");
$nodelist=$xmldocument
                        >getElementsByTagName("item");
foreach ($nodelist as $item) {
      foreach ($item->childNodes as $node) {
      switch ($node->nodeName) {
           case "title":
           print $node->textContent."<br>";
           break;
      }
}
}
```

In sostanza recuperiamo il contenuto del ramo 'item' ma poi siamo costretti a ciclare fra i dati ottenuti tramite il metodo childNodes e ad effettuare un confronto tramite il metodo nodeName.
Abbiamo usato una struttura di tipo Case solo pensando ad evoluzioni future che potevano prevedere anche l'output del campo description per esempio, nel caso più semplice sarebbe stato possibile anche usare una struttura 'if'.
Il metodo proposto funziona certamente, ma è ancora un po scomodo. Esistono delle forti limitazioni. Ad esempio è impossibile recuperare i dati sulla base di un confronto. Non sarebbe stato possibile recuperare i soli campi tali che la data di pubblicazione fosse stata inferiore a una prefissata. Questo tipo di operazione avrebbe comportato comunque il recuperare tutti gli elementi, e di seguito effettuare un confronto.
Esiste un metodo alternativo a questo, piuttosto funzionale e rappresentato dalla classe domXpath.

XPATH

Xpath rappresenta una sorta di linguaggio SQL per l'XML. Consente di effettuare interrogazioni su un oggetto di tipo DomDocument con un linguaggio semplice, che consente però di scendere nel dettaglio dell'albero XML ed è dotato di strutture tali da consentirci di sviluppare interrogazioni adeguate. L'esempio precedente, migrato ad un approccio di tipo XPath potrebbe assomigliare a qualcosa del genere:

```
$xmldocument = new DOMDocument();
$xmldocument
  >load("http://www.theregister.co.uk/excerpts.rss");
$xp = new DOMXPath($xmldocument);
$nodelist = $xp->query("//rss/channel/item/title");
foreach ($nodelist as $node) {
      print $node->textContent." <br>";
}
```

Si intuisce facilmente che abbiamo creato un nuovo oggetto $xp di classe DOMXPath. Il costruttore della classe DOMXpath prende come parametro l'oggetto di classe DOMDocument che contiene il documento XML. Il metodo query della classe DOMXpath prende come parametro una 'query' scritta con un linguaggio appropriato e restituisce un oggetto di tipo DOMNodeList.
Nel nostro caso la query è molto semplice, abbiamo richiesto tutti gli elementi della gerarchia /rss/channel/item/title.
L'esempio che segue è leggermente più complesso e mostra alcune delle caratteristiche avanzate di XPath

```
$xmldocument = new DOMDocument();
$xmldocument
  >load("http://www.theregister.co.uk/excerpts.rss");
$xp = new DOMXPath($xmldocument);
$nodelist = $xp
  >query("//rss/channel/item[substring(pubDate,9,3)
                                        "Jan"]/title2006);
foreach ($nodelist as $node) {
      print $node->textContent."<br>";}
```

Notate che questa volta la query è formulata in modo tale che vengano recuperati solo gli elementi Title tali che il campo pubDate del ramo item sia uguale a Jan ovvero Gennaio.
XPath offre alcuni metodi eleganti e al contempo complessi per l'elaborazione delle query, non è compito di questo libro procedere oltre nell'analisi di XPath, tuttavia è importante conoscere quali sono le classi che PHP mette a disposizione per il parsing di documenti XML tramite XPath.

SCRIVERE UN DOCUMENTO XML

Per certi versi ci sono meno considerazioni da fare sulla tecnica di produzione di un documento XML rispetto al suo parsing. Ci sarebbe qual-

che considerazione in più da fare rispetto al linguaggio XML che determina la struttura di un file XML ma non è certo questo l'articolo che parla approfonditamente di XML. L'esempio è abbastanza semplice:

```
$xmldocument = new DOMDocument();
$item = $xmldocument->createElement("item");
$titleitem = $xmldocument->createElement("title");
$titletext=$xmldocument
                        >createTextNode("impariamo PHP");
$titleitem->appendChild($titletext);
$item->appendChild($titleitem);
$xmldocument->appendChild($item);
print $xmldocument->saveXML();
```

Si crea un nuovo oggetto di tipo DOMDocument, vengono creati i vari rami del documento tramite il metodo createElement, i vari elementi vengono riempiti con il metodo createTextNode e innestati gerarchicamente fra loro con il metodo appendChild. Infine tutto viene mandato in output tramite il metodo saveXML.
Il risultato di questo genere di operazione è una pagina XML contenente il seguente codice:

```
<?xml version="1.0"?>
<item>
        <title>impariamo PHP</title>
</item>
```

LA GESTIONE DEI FILE

Come per tutti i linguaggi di programmazione, la gestione dei file rappresenta un aspetto importante anche in PHP. In realtà, essendo PHP un linguaggio orientato al web, è necessario considerare un certo numero di problemi in più legati da un lato alla necessità di garantire la sicurezza, dall'altro a utilizzare un'unica interfaccia sia per la gestione di file locali che per la gestione di file remoti. In questo PHP offre una soluzione decisamente flessibile. Considerate il seguente spezzone di codice:

```
<?
    $filename="/home/jaco/ioprogrammo.txt";
    $handle = fopen($filename, "r");
?>
```

la funzione fopen crea un collegamento stretto tra un nome di file e uno stream. Tutte le successive operazioni sul file dovranno essere compiute sullo stream. L'elemento interessante di questo genere di tecnica è che fopen prende in input qualunque nome di file gestibile come stream da PHP. Per selezionare il tipo di stream è sufficiente indicare uno scheme:// prima del nome del file. Ad esempio

```
<?
$filename="http://www.ioprogrammo.it/ioprogram
                                        o.txt";
$handle = fopen($filename, "r");
?>
```

apre uno stream direttamente dal web.
L'altro particolare interessante da considerare è il flag che segue l'argomento nomefile nella: funzione fopen. Questo flag indica la modalità con cui lo stream viene aperto. In particolare

'r'	Apre uno stream in sola lettura.
'r+'	Apre uno stream in lettura e scrittura
'w'	Apre uno stream in scrittura se il file non esiste lo crea
'w+'	Apre uno stream in lettura e scrittura, se il file non esiste lo crea
'a'	Apre uno stream in modalità ?Append?
'a+'	Apre uno stream in lettura e scrittura in modalità append
'x'	Crea un file e si posiziona in modalità scrittura
x+'	Crea un file e si posizione in modalità lettura e scrittura

Nell'interpretare questi flag è importante comprendere il concetto di "testina di lettura" o "cursore".
Immaginate uno stream come un nastro di dimensione finita steso su una superficie piana.
Per leggere il contenuto del nastro avrete bisogno di una testina posizionata su un binario che scorre lungo il nastro. La testina può essere posizionata all'inzio del nastro, alla fine, o una posizione qualunque del nastro, ma comunque dovrà scorrere in senso orizzontale da un lato o dall'altro senza mai staccarsi dai suoi binari.
Va da se che aprire un file in sola lettura significa posizionare la testina all'inizio del nastro e predisporla alla lettura dei contenuti.
Viceversa aprire un file in modalità append significa posizionare la testina alla fine del nastro, è poi possibile scrivere nuovi dati, ovvero appendere nuovi spezzoni di nastro a uno già esistente. Allo stesso modo ad esempio la modalità +w, apre .un file in lettura, se il file non esiste ne crea uno nuovo e pone la sua dimensione a zero e la testina all'inizio del nastro, se il file esiste, semplicemente pone a zero la sua dimensione e si predispone alla scrittura. Dunque è piuttosto importante utilizzare la modalità corretta se non si vuole incorrere nel rischio di cancellare qualche dato. Nei prossimi numeri approfondiremo altri aspetti della gestione del file system

IO PROGRAMMO BY EXAMPLE

IMPARA A PROGRAMMARE IN MODO PRATICO E DIVERTENTE, CON GLI ESEMPI PASSO PASSO CHE TI GUIDANO ALLA COSTRUZIONE DEL CODICE

Come contattarci?

Alla nostra redazione arriva spesso un considerevole numero di email riguardante temi specifici. Per consentirci di rispondere velocemente e in modo adeguato alle vostre domande abbiamo elaborato una FAQ – Frequently Ask Question – o risposte alle domande frequenti in italiano, di modo che possiate indirizzare le vostre richieste in modo mirato.

Problemi sugli abbonamenti

Se la tua domanda ha a che fare con una delle seguenti:

- Vorrei abbonarmi alla rivista, che devo fare?
- Sono un abbonato e non ho ricevuto la rivista, a chi devo rivolgermi?
- Sono abbonato ma la posta non mi consegna regolarmente la rivista, a chi devo rivolgermi?

Contatta **abbonamenti@edmaster.it** specificando che sei interessato a ioProgrammo. Lascia il tuo indirizzo email e indica il numero dal quale vorresti far partire l'abbonamento. Verrai contattato al più presto. Oppure puoi chiamare lo **02 831212**

Problemi sugli allegati

Se riscontri un problema del tipo:

- Ho acquistato ioProgrammo ed il Cdrom al suo interno non funziona. Chi me lo sostituisce?
- Ho acquistato ioProgrammo ma non ho trovato il cd/dvd all'interno, come posso ottenerlo?
- Vorrei avere alcuni arretrati di ioProgrammo come faccio?

Contatta **servizioclienti@edmaster.it**
Non dimenticare di specificare il numero di copertina di ioProgrammo e la versione: con libro o senza libro. Oppure telefona allo **02 831212**

Assistenza tecnica

Se il tuo problema è un problema di programmazione del tipo:

- Come faccio a mandare una mail da PHP?
- Come si instanzia una variabile in c++?
- Come faccio a creare una pagina ASP.NET

o un qualunque altro tipo di problema relativo a tecniche di programmazione, esplicita la tua domanda sul nostro forum: **http://forum.ioprogrammo.it**, uno dei nostri esperti ti risponderà. Le domande più interessanti saranno anche pubblicate in questa rubrica.

Problemi sul codice all'interno del CD

Se la tua domanda è la seguente

- Non ho trovato il codice relativo all'articolo all'interno del cd

Consulta la nostra sezione download all'indirizzo **http://cdrom.ioprogrammo.it**, nei rari casi in cui il codice collegato ad un articolo non sia presente nel cdrom, senza dubbio verrà reso disponibile sul nostro sito.

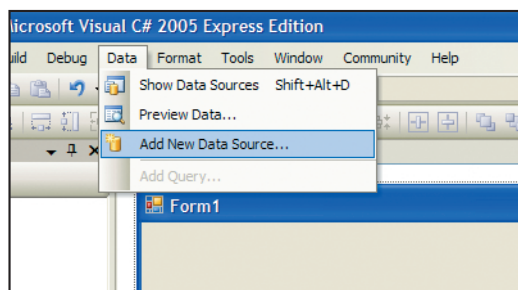
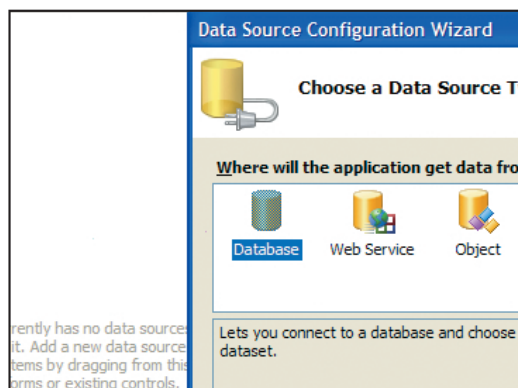
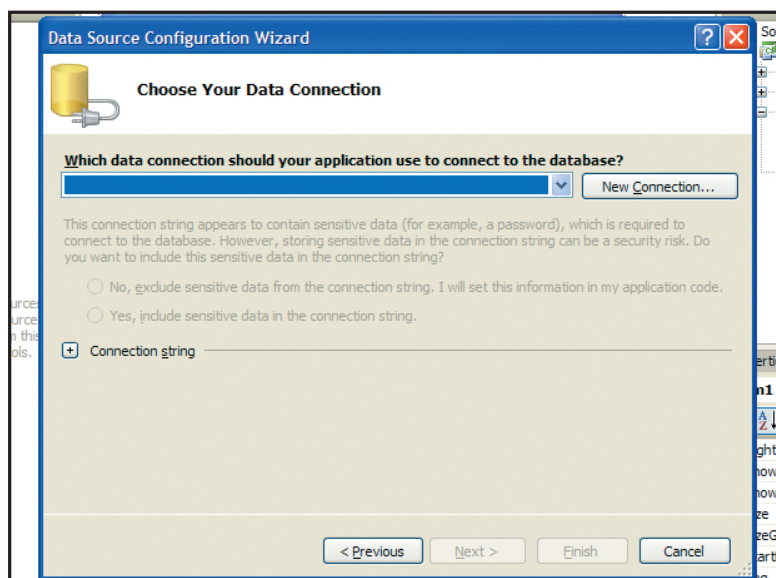
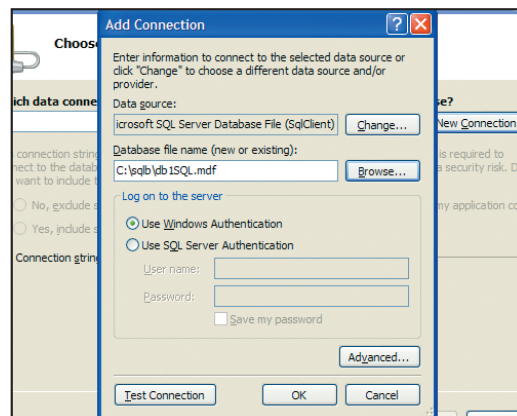
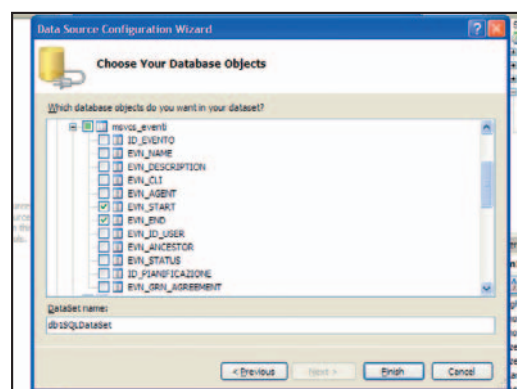
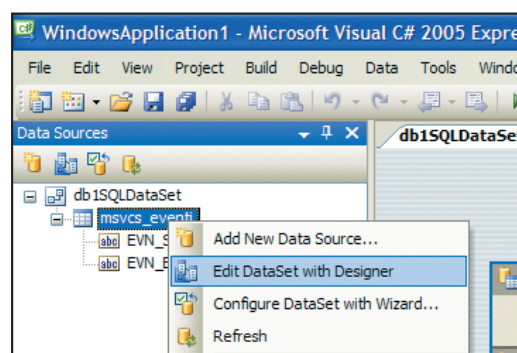
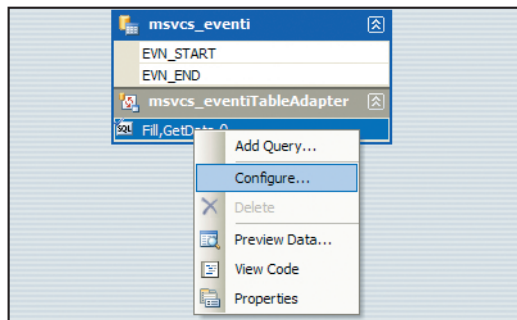
Idee e suggerimenti

Se sei un programmatore esperto e vuoi proporti come articolista per ioProgrammo, oppure se hai suggerimenti su come migliorare la rivista, se vuoi inviarci un trucco suggerendolo per la rubrica tips & tricks invia una email a **ioprogrammo@edmaster.it**

COME POSSO CONVERTIRE UNA DATA

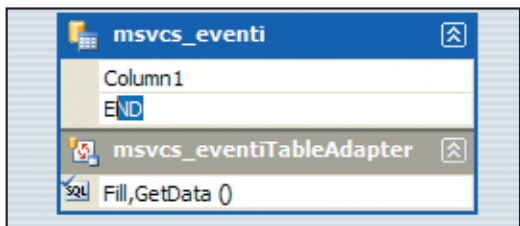
SUPPONIAMO DI DISPORRE UNA TABELLA CHE CONTIENE UNA COLONNA DATA_INIZIO E UNA DATA_FINE. TUTTE E DUE LE DATE SONO SALVATE IN FORMATO TIMESTAMP UNIX, ECCO COME VISUALIZZARLE IN UNA GRIDVIEW

VISUAL STUDIO

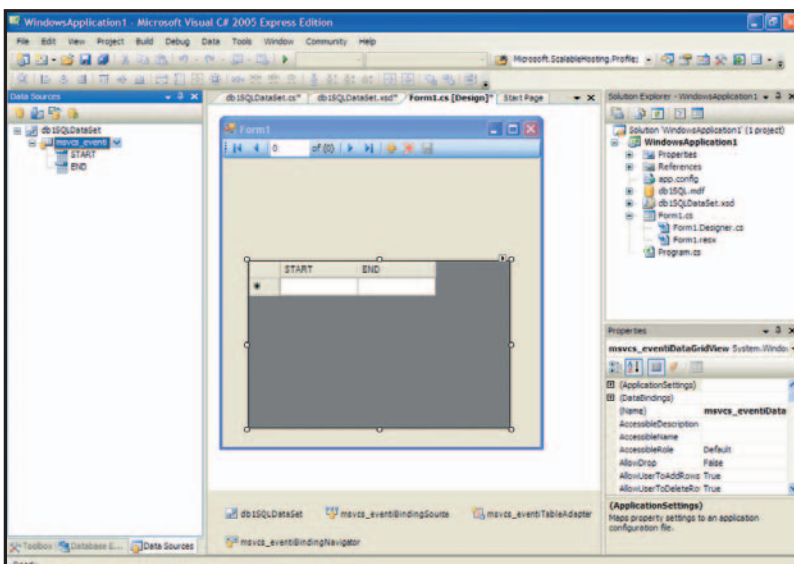
1 Creiamo un nuovo progetto è aggiungiamo subito una nuova "DataSource" dal menu "Data/Add New Data Source"

2 Dalla maschera che segue scegliamo "DataBase"

3 Scegliamo "New Connection" per scegliere la fonte da cui prelevare i nostri dati

4 Nella maschera che segue scegliamo il database a cui connettersi e andiamo oltre. Clicchiamo sempre next oppure yes, fino a quando non ci viene richiesto di scegliere i dati da importare

5 Selezioniamo la tabella e i dati che ci servono e andiamo avanti

6 Facciamo Click con il bottone destro del mouse nei "DataSources" e scegliamo "Edit DataSet with Designer"

7 Clicchiamo con il tasto destro del mouse sulla tabella che contiene i campi in formato Unix TimeStamp e selezioniamo "Configure"

8 Nella maschera che ci consente di modificare la query, cambiamo i valori standard come segue

SELECT dateadd(ss,cast(EVN_START as integer),'19700101'), dateadd(ss,cast(EVN_END as integer),'19700101') FROM dbo.msvcs_eventi

e completiamo il wizard cliccando su next fino alla fine

9 Rinominiamo i campi direttamente dal Query Designer semplicemente selezionandoli e digitando un nuovo nome

CHE COS'È IL FORMATO UNIX TIMESTAMP

È un formato piuttosto diffuso soprattutto nei database ti tipo mysql che spesso alimentano grandi progetti in rete. In sostanza esprime la data attuale calcolandola come differenza in numero di secondi con il primo gennaio 1970.

10 Ritorniamo alla form e trasciniamo direttamente dal data sources la tabella che ci interessa su di essa. Se tutto è andato a buon fine, mandando in esecuzione il programma vedremo il dato come ci interess

COME FUNZIONA

Abbiamo utilizzato funzioni specifiche di conversione di SQL Server. Prima effettuando un cast da stringa a intero della data espressa come Unix TimeStamp e poi utilizzando la funzione dateadd

COME POSSO CONCATENARE LE STRINGHE DI PERCORSO?

In C# un percorso valido di files e directory è rappresentato mediante stringhe di testo strutturate, nelle quali la distinzione tra le directory e tra directory e file avviene tramite il carattere separatore '\'. Ma bisogna stare attenti; scrivendo un percorso nel seguente modo

```
System.IO.FileInfo file=new
System.IO.FileInfo("C:\dir\esempio.txt");
```

l'interprete segnalerà una "Unrecognized escape sequence", cioè una sequenza di escape non riconosciuta. Per far riconoscere correttamente il percorso è necessario usare il doppio '\\'.

```
String s="c:\\dir\\esempio.txt";
```

oppure anteporre il carattere @ alla definizione della stringa

```
String s=@"c:\dir\esempio.txt";
```

Mediante il metodo statico Combine della classe Path è possibile tralasciare i controlli sul carattere separatore; sarà sufficiente passare al metodo le due stringhe per ottenere una nuova stringa percorso nata come concatenazione delle due.

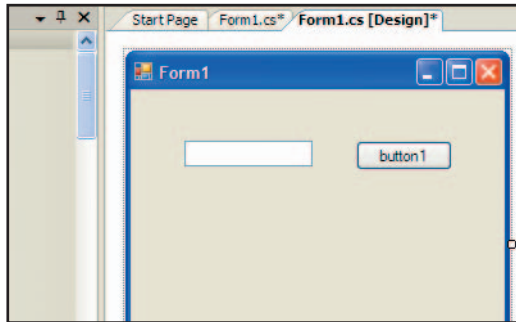
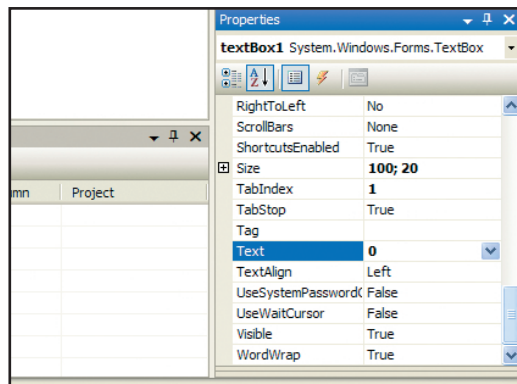
```
String s1=@"c:\dir";
String s2=@"sottodir1\esempio.txt";
String s3=
System.IO.Path.Combine(s1,s2); //
@"c:\dir\sottodir1\esempio.txt"
```

questo secondo metodo è sicuramente preferibile ai precedenti

CHE SIGNIFICA PASSARE UNA VARIABILE COME RIFERIMENTO?

SPESSO SENTIAMO PARLARE DI PASSAGGIO DI VARIABILI PER RIFERIMENTO O PER VALORE, MA CHE VUOL DIRE?

C#

1 Creiamo un nuovo progetto C# e nella form trasciniamo un bottone e un textbox

2 Poniamo la property Text della tesxtbox uguale a 0

ACCORGIMENTI NEL PASSAGGIO

Ai metodi che accettano parametri referenziati possono essere passati solo variabili e quindi non accetteranno espressioni costanti. Inoltre, le variabili di tipo ref dovranno essere anche inizializzate prima di essere passate come parametro.
La dichiarazione ref e out va specificata sia nella dichiarazione e sia nella chiamata del metodo.

Facciamolo in Visual Basic.NET
1) I passi dall'uno al tre sono identici. Invece il codice di gestione diventa il seguente:

```
Private Sub
    IncrementaContatore(ByRef
        MyValue As Integer)
    MyValue = MyValue + 1
End Sub
Private Sub Button1_Click(ByVal
sender As System.Object, ByVal e
    As System.EventArgs) Handles
        Button1.Click
    Dim MyValue As Integer
    MyValue =
    Convert.ToInt16(TextBox1.Text)
    IncrementaContatore(MyValue)
    TextBox1.Text =
        MyValue.ToString
End Sub
```

3 Clicchiamo due volte su button1 per generare il codice di gestione dell'evento click, e aggiungiamo il seguente codice

```
private void button1_Click(object sender,
                           EventArgs e)
    {
        int MyValue =
                Convert.ToInt16(textBox1.Text);
        IncrementaContatore(MyValue);
        textBox1.Text = MyValue.ToString();
    }
```

4 Poco più sopra aggiungiamo il metodo IncrementaContatore come segue

```
private void IncrementaContatore(int MyValue)
    {
        MyValue += 1;
    }
```

5 Mandiamo in esecuzione il tutto con la combinazione di tasti CTRL+SHIFT+B. Il programma non incrementa nulla. Le variabili passate per valore esistono solo nel corpo del metodo che le gestisce. Proviamo dunque a riscrivere il tutto passando i parametri come riferimento

```
private void IncrementaContatore(ref int MyValue)
    {
        MyValue += 1;
    }
private void button1_Click(object sender,
                           EventArgs e)
  {
     int MyValue = Convert.ToInt16(textBox1.Text);
        IncrementaContatore(ref MyValue);
        textBox1.Text = MyValue.ToString();
  }
```

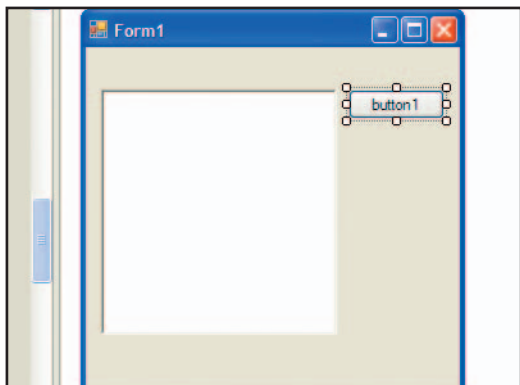
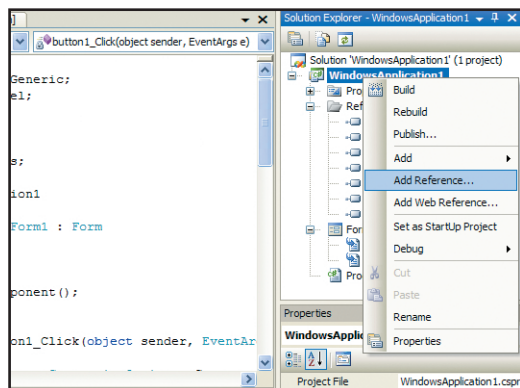
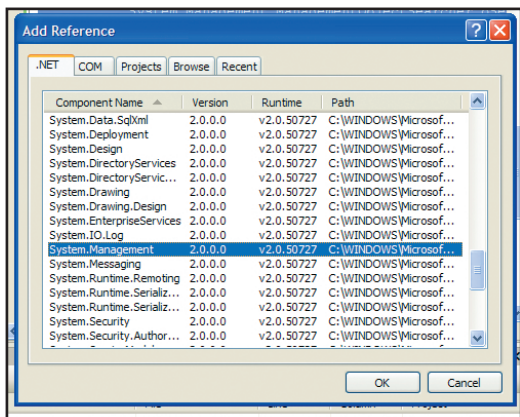
COME FUNZIONA

Una variabile A dichiarata nel metodo M1 è diversa dalla variabile A utilizzata nel metodo M2. Passare una variabile come riferimento stabilisce una sorta di legame fra il valore della variabile passata e quella usata nel metodo, cosa che consente di modificare il valore all'esterno.

COME POSSO OTTENERE INFORMAZIONI SULLO SPAZIO LIBERO NEI DISCHI?

E' POSSIBILE UTILIZZARE LA WINDOWS MEDIA INSTRUMENTATION, CHE CI RESTITUISCE UN'INFORMAZIONE ABBASTANZA PRECISA. VEDIAMO COME.

IN C#

1 Aggiungiamo alla form una richtextbox e un bottone

2 Clicchiamo con il tasto destro del mouse nel solution explorer e selezioniamo "Add Reference"

3 Scegliamo "System Management" e clicchiamo su Ok.

4 Clicchiamo due volte sul bottone e aggiungiamo il codice di gestione dell'evento click

```
private void button1_Click
                    (object sender, EventArgs e)
    {
        ConnectionOptions oConn = new
                        ConnectionOptions();
        //oConn.Username="";
        //oConn.Password="";
        ManagementScope oMs = new
    ManagementScope(@"\\MMachine\root\CIMV2",
                                    oConn);
        ObjectQuery oQuery = new
    ObjectQuery("select FreeSpace,Size,Name from
        Win32_LogicalDisk where DriveType=3");
        ManagementObjectSearcher oSearcher =
    new ManagementObjectSearcher(oMs, oQuery);
        ManagementObjectCollection
            oReturnCollection = oSearcher.Get();
        foreach (ManagementObject oReturn in
                            oReturnCollection)
        {

richTextBox1.Text+="Disco"+oReturn["Name"].ToSt
                                    ing()+"\n";
            richTextBox1.Text+="Spazio Libero: " +
            oReturn["FreeSpace"].ToString()+"\n";
            richTextBox1.Text += "Spazio: " +
                oReturn["Size"].ToString() + "\n\n";
        }
    }
```

VISUAL BASIC

1 i passi da uno a tre rimangono identici a quelli utilizzati in C#. Il codice di gestione dell'evento click sul bottone diventa invece

```
Private Sub Button1_Click(ByVal sender As
         System.Object, ByVal e As System.EventArgs)
                               Handles Button1.Click
        Dim oConn As ConnectionOptions
        Dim oQuery As ObjectQuery
        Dim oReturnCollection As
                         ManagementObjectCollection
        'oConn.Username="";
        'oConn.Password="";
        Dim oReturn As
ManagementObject
```

C#

VISUAL BASIC

I NAMESPACE DA USARE

In alto nel codice è necessario aggiungere

```
using
System.Management;
```

I NAMESPACE DA IMPORTARE IN VB

All-inizio del codice è necessario aggiungere:

```
Imports
    System.Management
```

```
Dim oMs As ManagementScope = New
ManagementScope("\\MMachine\root\CIMV2", oConn)
        oQuery = New ObjectQuery("select
FreeSpace,Size,Name from Win32_LogicalDisk
                            where DriveType=3")
Dim oSearcher As ManagementObjectSearcher =
New ManagementObjectSearcher(oMs, oQuery)
oReturnCollection = oSearcher.Get()
        For Each oReturn In oReturnCollection
            RichTextBox1.Text += "Disco" +
                oReturn("Name").ToString() + Chr(13)
            RichTextBox1.Text += "Spazio Libero: " +
                oReturn("FreeSpace").ToString() + Chr(13)
        RichTextBox1.Text += "Spazio: "
+ oReturn("Size").ToString() + Chr(13) + Chr(13)
        Next oReturn
```

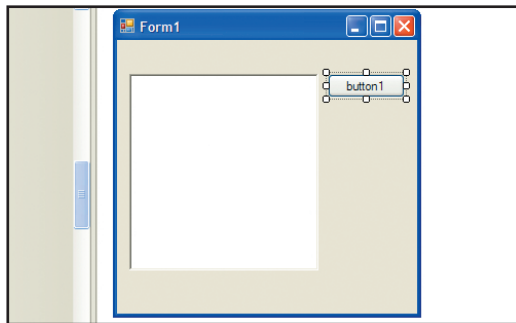
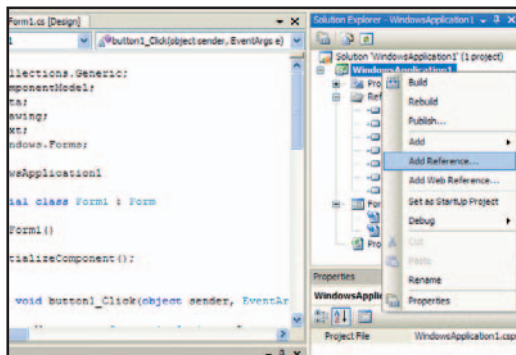
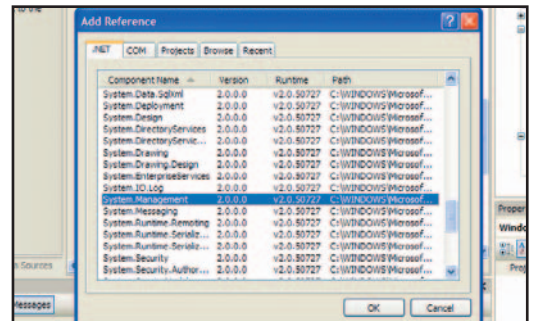
COME FUNZIONA

Prima di tutto ci colleghiamo alla macchina su cui effettuare la ricerca. In questo caso le credenziali di autenticazione sono commentate, in quanto l'utente locale è autorizzato a lavorare sulla macchina dove il programma è stato lanciato. In caso di macchina remota è opportuno inserire le credenziali di autenticazione corrette. Poi inseriamo lo scope ovvero \root\CIMV2. Eseguiamo una query in linguaggio WQL per ottenere le informazioni che ci servono da WMI. La query viene eseguita dall'oggetto ManagementObjectSearcher. Instanziamo un oggetto di tipo managementobjectcollection e riempiamolo tramite un Get. Il resto è una semplice iterazione attraverso la collection.

COME POSSO OTTENERE UN ELENCO DEI PROCESSI ATTIVI?

ANCHE IN QUESTO CASO CI VIENE INCONTRO LA WINDOWS MANAGEMENT INSTRUMENTATION, VEDIAMO COME:

FACCIAMOLO IN C#

1 Aggiungiamo alla form una textbox e un bottone

2 Clicchiamo sul solution explorer e poi di seguito su add reference

3 Scegliamo "System.Management" e clicchiamo su ok

4 Clicchiamo due volte sul bottone per aggiungere il codice di gestione dell'evento OnClick

```
private void button1_Click(object sender,
                                EventArgs e)
        {
ConnectionOptions oConn = new
                          ConnectionOptions();
            //oConn.Username="";
            //oConn.Password="";
            ManagementScope oMs = new
ManagementScope(@"\\ffarnesi\root\CIMV2",oC
                                       nn);
            ObjectQuery oQuery = new
      ObjectQuery("Select * from Win32_Process");
            ManagementObjectSearcher oSearcher
= new
ManagementObjectSearcher(oMs, oQuery);
            ManagementObjectCollection
                 oReturnCollection = oSearcher.Get();
```

```
            foreach (ManagementObject oReturn in
                                        oReturnCollection)
            {
                richTextBox1.Text+="Nome
            processo"+oReturn["Name"].ToString()+"-";
                richTextBox1.Text += "ProcessID" +
oReturn["ProcessId"].ToString() + "\n";
if (oReturn["Priority"] != null)
                    richTextBox1.Text += "Priorità: "
+ oReturn["Priority"].ToString()+"\n";

            }
        }
```

COME FUNZIONA

La teoria è identica a quella usata nel precedente articolo su come ricavare le informazioni su disco. Viene utilizzata la Windows Management Instrumentation, interrogata tramite il linguaggio WQL. Questa volta a cambiare e la query WQL che recupera l'elenco dei processi attivi. Il resto, è costituito da un ciclo sulla collection che contiene le informazioni desiderate

FACCIAMOLO IN VISUAL BASIC

1 I passi da uno a tre rimangono praticamente identici. Il codice di gestione dell'evento OnClick diventa:

```
  Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                                    Handles Button1.Click
        Dim oConn As ConnectionOptions
        Dim oQuery As ObjectQuery
        Dim oReturnCollection As
                          ManagementObjectCollection
        'oConn.Username="";
        'oConn.Password="";
        Dim oReturn As ManagementObject
        Dim oMs As ManagementScope = New
         ManagementScope("\\ffarnesi\root\CIMV2",
                                                oConn)
        oQuery = New ObjectQuery("Select * from
                                        Win32_Process")
        Dim oSearcher As
                   ManagementObjectSearcher = New
           ManagementObjectSearcher(oMs, oQuery)
        oReturnCollection = oSearcher.Get()
        For Each oReturn In oReturnCollection
            RichTextBox1.Text += "Nome Processso"
             + oReturn("Name").ToString() + Chr(13)
            RichTextBox1.Text += "ProcessID: " +
           oReturn("ProcessId").ToString() + Chr(13)
            If Not oReturn("Priority") = Nothing
                                                 Then
                RichTextBox1.Text += "Spazio: " +
           oReturn("Priority").ToString() + Chr(13) +
                                              Chr(13)
            End If
        Next oReturn
    End Sub
```

I NAMESPACE DA IMPORTARE PER C#

E' necessario aggiungere in testa al codice:

```
using
   System.Management;
```

I NAMESPACE DA IMPORTARE IN VISUAL BASIC

E' necessario aggiungere in testa al codice:

```
Imports
    System.Management
```

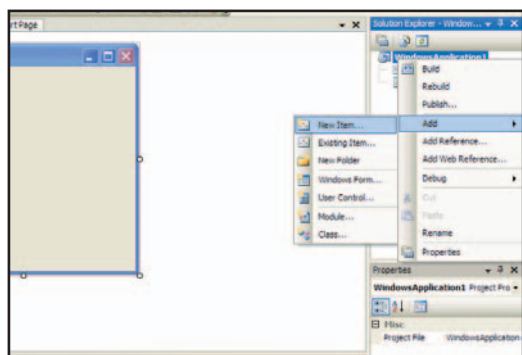
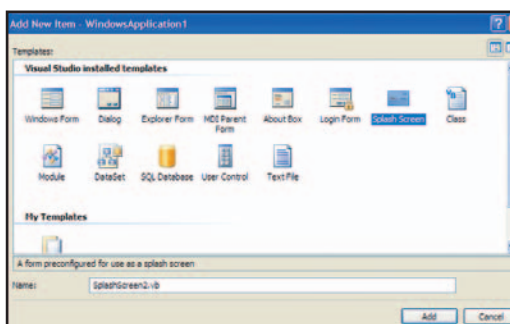
COME CREARE UNO SPLASHSCREEN PER UN'APPLICAZIONE?

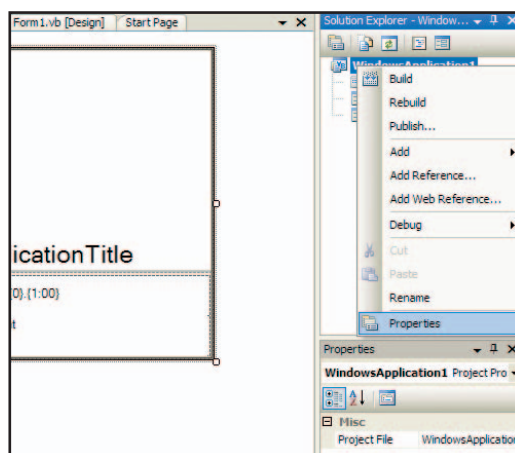
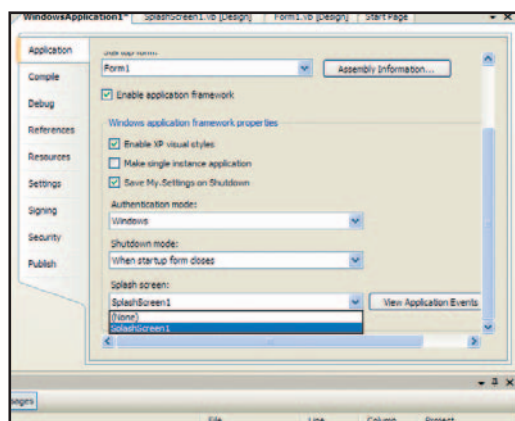
UNO SPLASHSCREEN È TIPICAMENTE UNA FORM CHE VIENE MOSTRATA ALL'UTENTE MENTRE ASPETTA CHE L'APPLICAZIONE VENGA CARICATA. VEDIAMO COME CREARLA

VISUAL BASIC

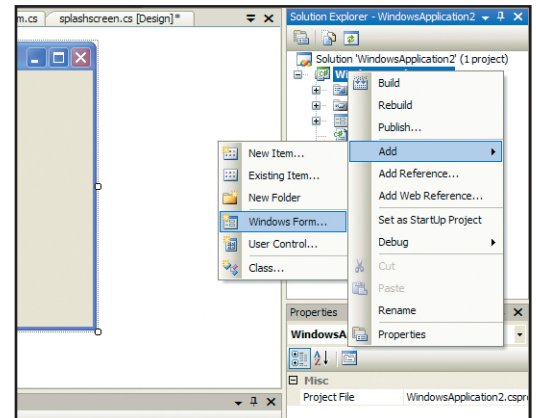
C#

FACCIAMOLO IN VISUAL BASIC

1 Creiamo un nuovo progetto Visual Basic. Una volta fatto clicchiamo con il tasto destro nel solution exporer e selezioniamo "Add/New Item"

2 Dalla finestra di dialogo che compare scegliamo "Splash Screen" e clicchiamo su ok

3 Apportiamo le modifiche grafiche che ci interessano alla form che rappresenta lo splashscreen

4 Clicchiamo con il tasto destro nel solution explorer alla voce che rappresenta l'applicazione e scegliamo properties

5 Dalla finestra che compare scegliamo "Splashscreen1" alla voce "Splash Screen"

Compiliamo e lanciamo l'applicazione. Se tutto è andato a buon fine vedremo prima lo splash screen e poi la mainform dell'applicazione.

FACCIAMOLO IN C#

1 Creiamo un nuovo progetto C#, clicchiamo con il tasto destro nel project exporer e poi su "Add Windows Forms". In C# non disponiamo di un wizard, quindi dovremo procedere manualmente

2 Applichiamo tutte le modifiche grafiche che contribuiranno a formare lo splashscreen

3 Clicchiamo due volte nel solution explorer su "program.cs" e nella finestra del codice che si aprirà aggiungiamo:

```
static void DoSplash()
        {  form2 sp = new form2();
           sp.ShowDialog() }
        static void Main()  {
           Application.EnableVisualStyles();
Application.SetCompatibleTextRenderingDefault
                                        (false);
           Thread th = new Thread(new
                          ThreadStart(DoSplash));
           th.Start();
           Thread.Sleep(3000);
           th.Abort();
           Thread.Sleep(1000);
           Application.Run(new Form1());
        }
```

COME FUNZIONA

In Visual Basic 2005 abbiamo usato un template, disponibile direttamente nell'ambiente e settato una proprietà dell'applicazione. In C# prima del lancio dell'applicazione abbiamo instanziato un thread al cui interno gira il metodo DoSplash che non fa altro che far vedere lo splashscreen. Quando viene richiamato l'abort del thread anche lo splashscreen viene chiuso e immediatamente dopo l'applicazione viene lanciata.

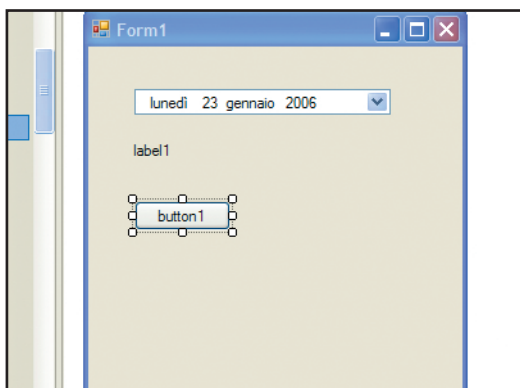
COME POSSO SAPERE QUANTI GIORNI MANCANO AD UNA CERTA DATA?

UN CONTROLLO UTILE PER EFFETTUARE UN COUNTDOWN, O PER INVIARE UN AVVERTIMENTO A TEMPO SCADUTO, VEDIAMO COME REALIZZARLO

C#

VISUAL BASIC.NET

FACCIAMOLO IN C#

1 1) Su una form posizioniamo un componente di tipo DateTimePicker, un bottone e una label

2 Clicchiamo due volte sul bottone per inserire il codice di gestione dell'evento OnClick. Il codice da inserire sarà il seguente

```
private void button1_Click(object sender,
                                     EventArgs e)
    {
        DateTime datafinale = new DateTime();
        datafinale = dateTimePicker1.Value;
        TimeSpan differenza = new TimeSpan();
        differenza =
              datafinale.Subtract(DateTime.Now);
        Int32 giorni  = differenza.Days;
        Int32 ore = differenza.Hours;
        Int32 minuti = differenza.Minutes;
        label1.Text =
```

```
  giorni.ToString()+":"+ore.ToString()+":"+minuti.To
                                            tring();
}
```

COME FUNZIONA?

Viene sfruttata la classe TimeStamp che rappresenta un intervallo di tempo fra due date. L'intervallo viene ricavato utilizzando il metodo substract della classe DateTime. L'informazione viene mandata in output su una label concatenando le stringhe che nel timestamp rappresentano giorni, ore, minuti.

FACCIAMOLO IN VISUAL BASIC

1 Il passo uno rimane identico a quello effettuato in C#, il codice da inserire in relazione all'evento OnClick sul bottone diventa invece

```
Private Sub Button1_Click(ByVal sender As
        System.Object, ByVal e As System.EventArgs)
                              Handles Button1.Click
    Dim datafinale As DateTime = New DateTime
    datafinale = DateTimePicker1.Value
    Dim differenza As TimeSpan =
              datafinale.Subtract(DateTime.Now)
    Dim giorni As Integer = differenza.Days
    Dim ore As Integer = differenza.Hours
    Dim minuti As Integer = differenza.Minutes
    label1.Text =
  giorni.ToString()+":"+ore.ToString()+":"+minuti.To
                                            tring()
  End Sub
```

COME OTTENERE INFORMAZIONI AGGIUNTIVE NELLA STATUS BAR?

LO SCOPO È VISUALIZZARE UNA STRINGA NELLA STATUS BAR AL PASSAGGIO DEL MOUSE SOPRA UN LINK, VEDIAMO COME:

JAVASCRIPT

1 Aggiungiamo il link nella pagina html come segue:

```
<a href=http://
www.ioprogrammo.it">ioProgrammo</A>
```

questo sarà il nostro punto di partenza

2 Modifichiamolo aggiungendo un gestore degli eventi

```
<a href="http://www.ioprogrammo.it"
      onmouseover="window.status='Home page di
                    IoProgrammo'; return true;"
                 OnMouseOut="window.status=''">
```

```
IoProgrammo </a>
```

COME FUNZIONA?

E' abbastanza semplice, al passaggio del mouse sul link viene richiamato l'evento onmouseover che non fa altro che modificare la stringa all'interno della status bar. All'uscita del mouse dal focus del link, il valore della stringa nella status bar viene svuotato.

COME POSSO FARE IN MODO DI ESEGUIRE UN SUONO AL PASSAGGIO DEL MOUSE SU UN'IMMAGINE?

L'IDEA È QUELLA DI FORNIRE AGLI UTENTI UNA PREVIEW DI UN BRANO MUSICALE AL PASSAGGIO DEL MOUSE SU UNA COPERTINA DI UN ALBUM PER ESEMPIO, VEDIAMO COME:

JAVASCRIPT

1 Inserite il seguente codice all'interno della pagina html

```
<bgsound src="#" id="mycd" autostart="true">
<br /><br />
<bgsound src="#" id="mycd" autostart="true">
```

2 Utilizzate l'evento 'onmouseover' per richiamare lo spezzone di canzone ed onmouseout per interrompere quando il cursore viene posto fuori dalla immagine.

```
<img src="brano.gif"
onmouseover="document.getElementById('mycd').s
rc='alleluia.mid'"
onmouseout="document
        .getElementById('mycd').sc=''">
```

COME FUNZIONA

Viene posizionato un "placeholder" di tipo bgsound nella pagina, con il valore autostart settato a true, inizialmente questo elemento sarà valorizzto con un # quindi punterà ad un valore vuoto. Al passaggio del mouse sull'immagine bgsound verrà valorizzato con il corrispondente brano. Allo stesso modo uscendo dal campo d'azione dell'immagine il valore di bgsound sarà riportato a null

COME POSSO CREARE UN BOTTONE DI FORMA CIRCOLARE?

CREIAMO UNA NUOVA CLASSE DERIVANDOLA DA UNA PRECEDENTE E OTTENIAMO UN BOTTONE CON UN NOSTRO PRECISO LOOK

VISUAL BASIC.NET

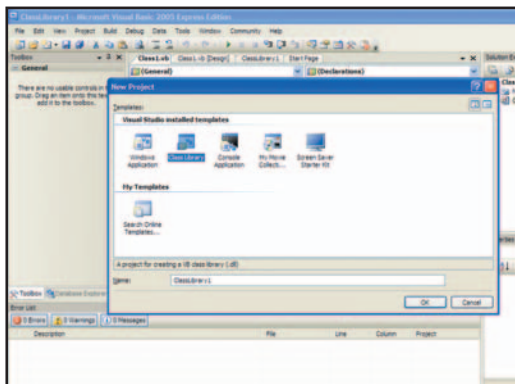
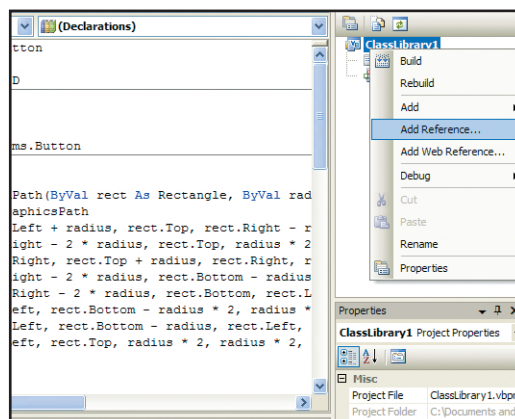
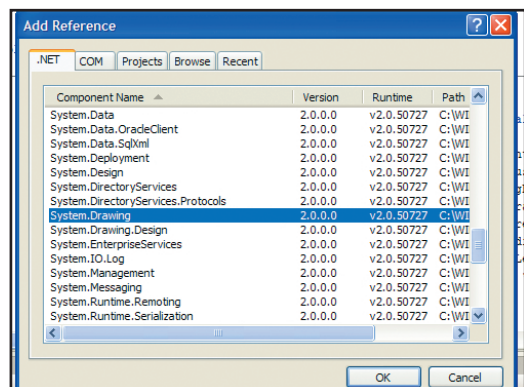
C#

I NAMESPACE DA IMPORTARE IN VISUAL BASIC

All'inizio del codice è necessario aggiungere le seguenti righe:

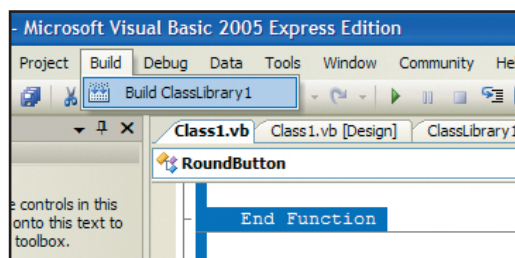
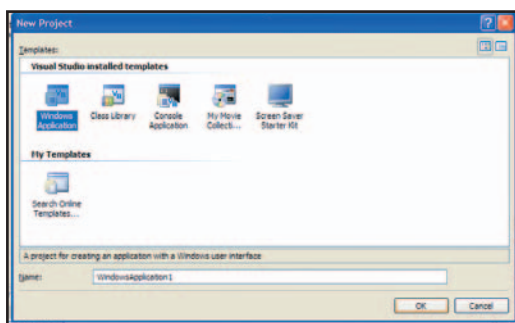
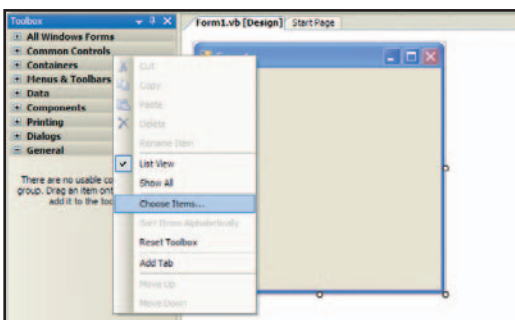
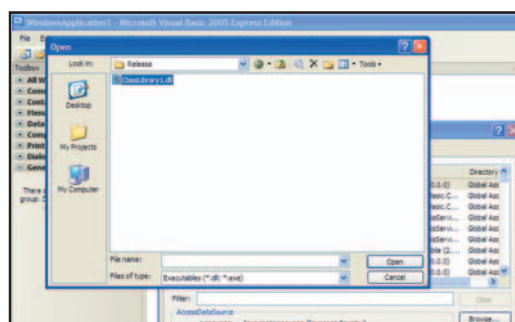
```
Imports System.Drawing
                Imports
System.Drawing.Drawin
                     2D
Imports
  System.Windows.Forms
```

FACCIAMOLO IN VISUAL BASIC

1 Creiamo un nuovo progetto di tipo class library

2 Clicchiamo con il tasto sinistro del mouse nel solution explorer e nel menu a tendina che compare su "Add Reference"

3 Cerchiamo “System.Drawing” e aggiungiamolo al progetto cliccando su Ok, allo stesso modo aggiungiamo anche il riferimento a System.-Windows.Form

4 Aggiungiamo il codice di gestione della classe

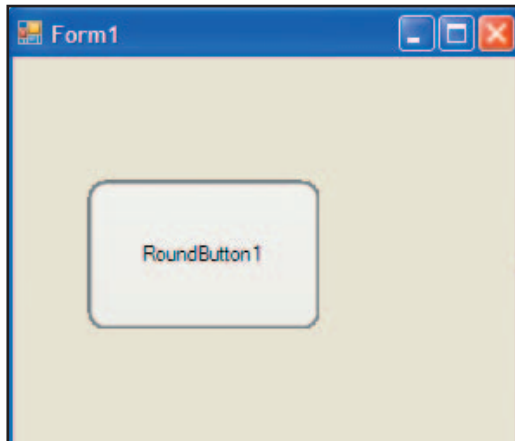
```
Public Class RoundButton
  Inherits System.Windows.Forms.Button
  Private Function GetRoundedPath(ByVal rect As
    Rectangle, ByVal radius As Int32) As GraphicsPath
      Dim roundRect As New GraphicsPath
      roundRect.AddLine(rect.Left + radius,
            rect.Top, rect.Right - radius, rect.Top)
      roundRect.AddArc(rect.Right - 2 * radius,
            rect.Top, radius * 2, radius * 2, 270, 90)
      roundRect.AddLine(rect.Right, rect.Top +
            radius, rect.Right, rect.Bottom - 10)
      roundRect.AddArc(rect.Right - 2 * radius,
   rect.Bottom - radius * 2, radius * 2, radius * 2, 0,
                                                   90)
      roundRect.AddLine(rect.Right - 2 * radius,
         rect.Bottom, rect.Left + radius, rect.Bottom)
      roundRect.AddArc(rect.Left, rect.Bottom -
            radius * 2, radius * 2, radius * 2, 90, 90)
      roundRect.AddLine(rect.Left, rect.Bottom -
            radius, rect.Left, rect.Top + radius)
      roundRect.AddArc(rect.Left, rect.Top, radius *
            2, radius * 2, 180, 90)
      Return roundRect
  End Function
  Protected Overrides Sub OnPaint(ByVal pe As
            System.Windows.Forms.PaintEventArgs)
      MyBase.OnPaint(pe)
      Dim newRectangle As Rectangle =
                        Me.ClientRectangle
      newRectangle.Inflate(-3, -3)
      Dim buttonPath As GraphicsPath =
            GetRoundedPath(newRectangle, 10)
      pe.Graphics.SmoothingMode =
                  SmoothingMode.AntiAlias
      pe.Graphics.DrawPath(New Pen(Color.Gray,
                        4), buttonPath)
      Me.Region = New
```

```
            System.Drawing.Region(buttonPath)
  End Sub
```

5 Compiliamo il tutto utilizzando il menu build

6 Creiamo un nuovo progetto di tipo “Windows Application”

7 Clicchiamo con il tasto destro del mouse nella toolboox e di seguito su “Choose Items”

8 Nella finestra che segue scegliamo “Browse” e navighiamo alla ricerca del percorso in cui abbiamo compilato il nostro nuovo componente, tipicamente “classlibrary1.dll”

I NAMESPACE DA IMPORTARE IN C#

All’inizio del codice relativo al nuovo componente è necessario utilizzare

```
using System.Drawing;
using
 System.Drawing.Drawin
                  g2D;
using
 System.Windows.Forms
                    ;
```

9 Infine clicchiamo su Ok e trasciniamo il nuovo componente sulla form

COME FUNZIONA

Abbiamo creato un nuovo componente effettuando derivandolo direttamente dalla classe Button di Windows Form. Abbiamo effettuato l'ovverride del metodo paint che appunto si occupa del disegno del bottone, sovrascrivendolo con un nostro proprio disegno grafico. Infine abbiamo importato il componente in un nuovo progetto e lo abbiamo usato in modo usuale

FACCIAMOLO IN C#

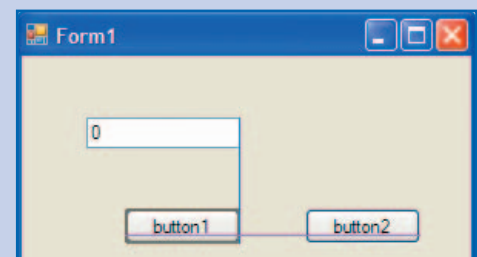
1 I passaggi dall'uno al tre rimangono identici. Il codice della classe invece diventa il seguente:

```
public class
        RoundButton:System.Windows.Forms.Button
{
        GraphicsPath GetRoundedPath(Rectangle
                                  rect,  Int32 radius)
{
                GraphicsPath roundRect = new
                                          GraphicsPath();
                roundRect.AddLine(rect.Left +
   radius, rect.Top, rect.Right - radius, rect.Top);
                roundRect.AddArc(rect.Right - 2
 * radius, rect.Top, radius * 2, radius * 2, 270, 90);
                roundRect.AddLine(rect.Right,
   rect.Top + radius, rect.Right, rect.Bottom - 10);
                roundRect.AddArc(rect.Right - 2
      * radius, rect.Bottom - radius * 2, radius * 2,
                                  radius * 2, 0, 90);
                roundRect.AddLine(rect.Right - 2
         * radius, rect.Bottom, rect.Left + radius,
                                        rect.Bottom);
                roundRect.AddArc(rect.Left,
 rect.Bottom - radius * 2, radius * 2, radius * 2, 90,
                                                 90);
                roundRect.AddLine(rect.Left,
  rect.Bottom - radius, rect.Left, rect.Top + radius);
                roundRect.AddArc(rect.Left,
          rect.Top, radius * 2, radius * 2, 180, 90);
                return roundRect;
  }

        override protected void
 OnPaint(System.Windows.Forms.PaintEventArgs pe)
  {
          base.OnPaint(pe);
                Rectangle newRectangle =
                                this.ClientRectangle;
                newRectangle.Inflate(-3, -3);
                GraphicsPath buttonPath  =
                     GetRoundedPath(newRectangle, 10);
                pe.Graphics.SmoothingMode =
                              SmoothingMode.AntiAlias;
                pe.Graphics.DrawPath(new
                   Pen(Color.Gray, 4), buttonPath);
                this.Region = new
                    System.Drawing.Region(buttonPath);
    }
}
```

2 Per aggiungere il nuovo componente alla toolbox si eseguono gli stessi passaggi utilizzati nei passi da 5 a 8 nel caso di visual basic

COME POSSO ALLINEARE I CONTROLLI SU UNA FORM?

Tipicamente si passa moltissimo tempo a sistemare l'interfaccia grafica. In Visual Studio 2005 il problema è stato risolto. È sufficiente usare le guide associate ad ogni controllo in verticale e in orizzontale come in figura. Le linee rosse indicano l'allineamnto. Le guide sono attivate automaticamente nell'ambiente, senza ulteriori configurazioni.

COME POSSO OTTENERE UN'IMMAGINE CON ANGOLI ARROTONDATI?

UTILE PER ADATTARE UN'IMMAGINE DA UN DATABASE AL LAYOUT DEL PROPRIO SITO WEB

PHP

Il trucco è abbastanza semplice. Prenderemo un'immagine, già presente sull'hard disk, vi applicheremo sopra, nei quattro angoli le rispettive immagini rappresentanti un bordo arrotondato. Vediamo come

FACCIAMOLO IN PHP

1 Scriviamo il prototipo di una funzione che prende come parametri i puntatori a due file di immagini. Il primo parametro rappresenterà l'immagine di partenza, il secondo l'immagine di destinazione, ossia quella con i bordi arrotondati

```
function creaimmagine
                        ($sorgente, $destinazione) {
}
```

2 Aggiungiamo il codice che ci serve per recuperare l'immagine di partenza e le informazioni sulle sue dimensioni

```
function creaimmagine
                        ($sorgente, $destinazione) {
$info = getimagesize($sorgente);
  switch ($info['mime']) {
   case 'image/jpeg' :
     $image = imagecreatefromjpeg($sorgente);
     break;
   case 'image/png' :
     $image = imagecreatefrompng($sorgente);
     break;
   case 'image/gif' :
     $image = imagecreatefromgif($sorgente);
     break;
   default:
     return FALSE;

  }
  $image_larghezza = imagesx($image);
  $image_altezza = imagesy($image);
}
```

3 settiamo il canale alfa in modo che quando incolleremo i vari angoli nei punti giusti il risultato dell'operazione non restituisca un'immagine opaca o dai colori falsati

```
imagealphablending($image, true);
```

4 Incolliamo i vari angoli nei punti giusti

```
// Overlay left top corner
 $crnimage_nw =
              imagecreatefrompng("crn_nw.png");
 $crnimage_nw_w = imagesx($crnimage_nw);
 $crnimage_nw_h = imagesy($crnimage_nw);
 imagecopy($image, $crnimage_nw, 0, 0, 0, 0,
    $crnimage_nw_w, $crnimage_nw_h);
```

e ripetiamo il procedimento per tutti gli angoli sistemando opportunamente le coordinate

5 infine riproduciamo la nuova immagine con un formato conforme all'originale

```
switch ($info['mime']) {
 case 'image/jpeg'
   imagejpeg($image, $destinazione, 100);
   break;
 case 'image/png' :
   imagepng($image, $destinazione);
   break;
 case 'image/gif' :
   imagegif($image, $destinazione);
   break;
```

COME FUNZIONA

Abbiamo sfruttato il modulo GD di php per leggere il contenuto dell'immagine sorgente, settare il canale alfa e incollarvi sopra 4 angoli arrotondati, sempre attraverso il modulo GD abbiamo creato l'output di una nuova immagine.

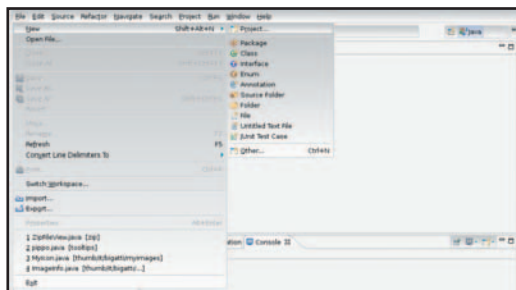
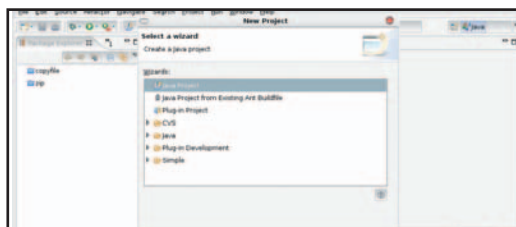
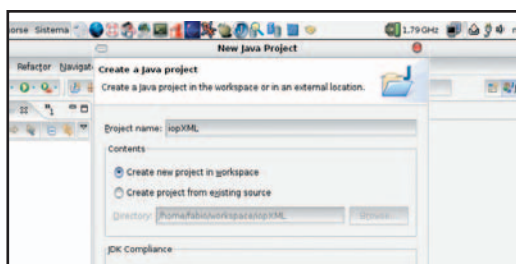
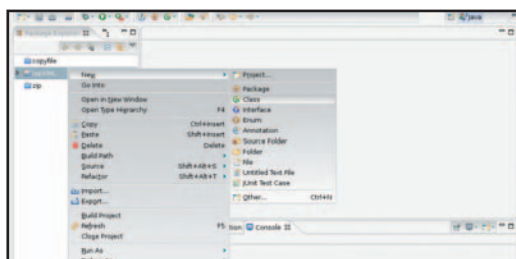
COSA SONO LE XFORMS?

Si tratta di un metodo alternativo per il passaggio dei dati fra pagina e pagina. Le informazioni viaggiano attraverso i vari link in formato XML invece del classico formato RAW_DATA. Da notare che non tutti i browser supportano adeguatamente le XFORMS. Firefox ed Explorer lo fanno in modo personalizzato.

COME POSSO EFFETTUARE IL PARSING DI UN FILE XML CON DOM

ECCO UN METODO CHE SFRUTTA IL DOCUMENT OBJECT MODEL PER OTTENERE UN OGGETTO A PARTIRE DA UN FILE XML

JAVA

1 Da eclipse iniziamo con il creare un nuovo progetto

2 dalla schermata che segue scegliamo "Java Project"

3 Diamo un nome al progetto, ad esempio iopXML, proseguiamo con il wizard, lasciandoinalterate le opzioni che ci vengono proposte, fino alla fine

4 Aggiungiamo una nuova classe al progetto, agendo con il tasto destro del mouse sul nome del progetto

5 Diamo un nome alla nuova classe, ad esempio XmlParserDOM, assicuriamoci anche di avere spuntato la checkbox per la creazione del main

6 aggiungiamo gli import necessari nelle prime righe del codice

```
import java.io.*;
import javax.xml.parsers.*;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;
```

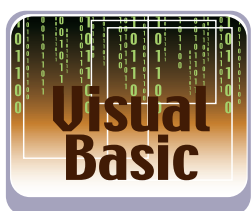
7 La classe modificata avrà il seguente aspetto

```
public class XmlParserDOM {
        String filename;
        Document document;
        public XmlParserDOM( String filename ) {
                this.filename = filename;
        }
        public void parse() throws SAXException,
                                        IOException,
        ParserConfigurationException {
        DocumentBuilderFactory factory=
DocumentBuilderFactory.newInstance();
factory.setValidating(false);
document =factory.newDocumentBuilder().parse(
                                newFile(filename)
                );
        }
        public Document getDocument() {
                return document;
        } }
```

COME FUNZIONA

Nel costruttore viene inizializzato il puntatore al file xml. Il metodo parse, ricava le informazioni sul Document e le conserva nella proprietà omonima. Infine il metodo getDocument restituisce il documento che avevamo richiesto

COME USARE SQL SERVER EXPRESS 2005

VI PRESENTIAMO ALCUNI STRUMENTI DELLA FAMIGLIA MICROSOFT EXPRESS ED INTRODUCIAMO UN'APPLICAZIONE CLIENT-SERVER CHE PERMETTE DI CATALOGARE DVD E IMMAGINI.

Microsoft SQL Server 2005 Express Edition appartiene alla famiglia dei prodotti Express lanciati da Microsoft con l'obiettivo di soddisfare le esigenze di sviluppatori abituati ad utilizzare strumenti free o OpenSource.
Insieme a SQL Express sono state rilasciate le versioni free dei principali Tool di Visual Studio 2005 tra i quali anche Visual Basic. SQL Express, anche se progettato sul dotNET Framework, può interagire con Visual Basic 6; anzi sono state introdotte nuove funzionalità che rendono l'interazione ancora più semplice e produttiva, e per certi versi, come vedremo, la rendono simile a quella con i database Access. Sul sito Microsoft sono disponibili tutti i Download necessari per il corretto funzionamento di SQL Server Express, gli strumenti ad esso collegati e suggerimenti per la loro corretta installazione. Se ancora non l'avete fatto, connettetevi al seguente link http://go.microsoft.com/fwlink/?LinkId=31401 e scaricate SQL Server Express e gli strumenti che vi consiglieremo nel corso dell'articolo. Naturalmente SQL Express può essere installato soltanto se è installata la versione 2.0 di dotNET Framework ed almeno Windows 2000. Per accedere da programma a SQL Express è stata introdotta la tecnologia di accesso ai dati nominata SQL Native Client. Questa è un'applicazione standalone, installata insieme a SQL Express, che può essere usata per accedere ai database tramite gli oggetti ADO-OLE DB e ODBC. Nell'articolo, dopo aver spiegato, come manipolare un database SQL Server, introdurremo un'applicazione che gestisce una mediateca. Questa si poggia su un database SQL Server, usa dei moduli di classe, delle query SQL e delle Stored Procedure e, tra l'altro, permette di gestire dei campi di tabelle in formato Image (che possono contenere file jpg, .bmp ecc.). Tra gli strumenti per SQL Server Express introdurremo SQL Server Management Studio Express, un free database Manager che permette di creare e manipolare i principali elementi di un database SQL Server. Data l'importanza e la vastità dell'argomento, la trattazione sarà distribuita su due articoli.

STORED PROCEDURE

Una Stored Procedure (SP) è uno script composto da un insieme di istruzioni SQL ed istruzioni per il controllo di flusso. Una SP è identificata attraverso un nome ed è elaborata come una singola unità, essa può essere richiamata, attraverso un Command ADO, da un'applicazione scritta in Visual Basic. La SP ai fini della programmazione può essere vista come una procedura o come una funzione. Infatti, può ricevere e restituire dei parametri (Parameter).

SQL SERVER 2005 EXPRESS

SQL Server 2005 Express, dalla Microsoft, è considerato "lo strumento più idoneo per sviluppare semplici applicazioni data driver" e per avvicinarsi a SQL Server 2005 che prevede altre tre versione (naturalmente a pagamento) e cioè la Workgroup, la Standard e la Enterprise. Senza dilungarci sulle altre tre, parliamo brevemente della Express. Essa è una versione completa che permette di gestire database fino a 4GB, per essa sono forniti, sempre gratis, un Management Tool e un Report Tool. Per quanto riguarda le caratteristiche, SQL Server 2005 è stato migliorato nelle tre aree principali: l'Enterprise Data Management, il Business Intelligence e lo Sviluppo. Per quanto riguarda quest'ultimo aspetto, si è lavorato soprattutto sull'integrazione con dotNET Framework e VS 2005, sul supporto nativo di XML, sull'interoperabilità con standard aperti e Web Services, sull'ottimizzazione delle connessioni attraverso il Multiple Recordsets (più Recordset sulla stessa connessione) e altri aspetti che potete approfondire scaricando SQL Server Books Online (120 MB d'informazioni). Inoltre, è stato sviluppato un completo set di strumenti grafici e a linea di comando, che permettono di programmare e amministrare le varie parti di SQL Server; per esempio permettono di ricavare

informazioni diagnostiche, importare, esportare e trasformare dati ecc.

SQL NATIVE CLIENT

SQL Native Client è la nuova tecnologia di accesso ai dati installata da SQL Express. Essa in un'unica libreria racchiude le caratteristiche del provider SQL OLE DB del driver ODBC e nuove funzionalità. Dato che SQL Native Client non dipende esplicitamente da una particolare versione di MDAC (Microsoft Data Access Components) è ancora possibile utilizzare la versione di MDAC installata sul vostro computer. Quindi si può accedere ad SQL Server 2005, anche, utilizzando direttamente le vecchie tecnologie di accesso, che però non permettono di utilizzare le nuove features quali XML data type, gli UDT (tipi definiti dall'utente) i multiple active result sets (MARS) ecc. Un'altra feature introdotta con SQL Native Client è legata al trasporto dei database, con SQL Express tutte le informazioni del database possono essere racchiuse e trasportate nel file MDF. Questo, sicuramente, rappresenta una rivoluzione per gli sviluppatori di Web Applications che potranno mantenere una copia del file MDF in una directory del sito ed utilizzarla come un file Access. Naturalmente anche in questi scenari SQL Server deve essere sempre disponibile.

CONNETTERSI AD UN DATABASE

In base a quanto illustrato nei paragrafi precedenti per connettere, a livello di codice, un'applicazione Visual Basic 6 ad un database SQL Express si possono utilizzare gli oggetti ADO indifferentemente, insieme ad OLE DB o insieme all'SQL Native Client. In particolare, nel corso dell'articolo, presentiamo tre modi per connettersi ad un database: connessione diretta al file MDF, utilizzo della classica ConnectionString ed utilizzo di un DSN. Ricordiamo che un DSN (Data Source Name) tramite i driver dell'ODBC crea una connessione ad una fonte dati (cioè ad un database). Nei nostri esempi utilizzeremo un DSN Utente nominato DSNdvd, connesso ad un database SQL Server, a sua volta nominato DVD. Naturalmente DSNdvd è impostato utilizzando il Driver SQL Native Client.
Con l'arrivo di SQL Native Client la formattazione della ConnectionString, della connessione ADO, è leggermente modificata. Ricordiamo che in generale una ConnectionString presenta le seguenti parti:

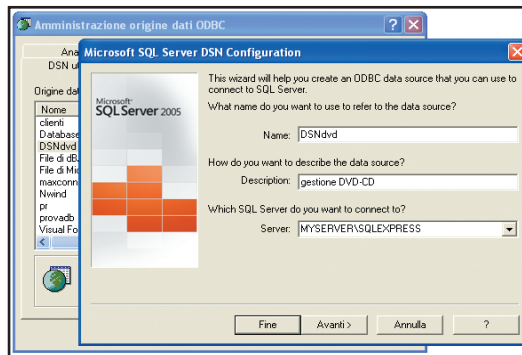



Fig. 1: ***La prima pagina del Wizard creazione DSN***

ConnectionString =“Provider=…;
Data Source =;
Integrated Security= …;
AttachDBFileName =…;…”

Con Provider si specifica il nome del Provider di dati utilizzato (controllare Tabella 1). Quando, invece, si utilizza un DSN, Data Source si imposta uguale al nome del DSN. Per utilizzare le credenziali di sicurezza di Windows, per accedere al database, si specifica Integrated Security=SSPI ecc. Rispetto alla versione precedente, la novità, nella stringa di connessione, è AttachDBFileName che, come accennato, permette di specificare il nome del file MDF che riproduce il database.

Provider	Descrizione
MSDataShape	Supporta la costruzione di oggetti Recordset gerarchici
MSDASQL	Consente a ADO di connettersi a qualsiasi origine dati ODBC
SQLOLEDB	Consente a ADO di accedere direttamente a Microsoft SQL Server
Microsoft.Jet. OLEDB.4.0	Con esso ADO può accedere ai database Access
SQLNCLI.1	Consente a ADO di accedere a SQL Server Express con SQL Native Client

Di seguito presentiamo due esempi di stringa di connessione. Nel successivo paragrafo presenteremo la procedura completa che permette di connettersi ad un file MDF.

```
a) ConnectionString = "Provider=SQLO
                LEDB;Integrated Security=SSPI;" & _
   "Server=serversqlexpress\sqlexpress;database=dvd"
```

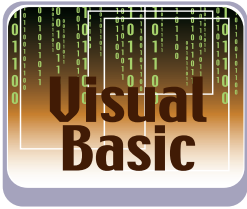
Questa serve per connettersi ad un database SQL Express con il vecchio provider SQLOLEDB.

```
b) ConnectionString = "Provider=SQLN
                CLI.1;Integrated Security=SSPI;" & _
        "AttachDBFileName=" & "vostropath" &
   "\dvd.MDF;Data Source= serversqlexpress\sqlexpress"
```

Questa stringa è utilizzata per connettersi, con il

NOTE

CONNECTION, COMMAND DI ADO 2

Riassumiamo le caratteristiche dei principali oggetti ADO. Un oggetto Connection rappresenta una connessione aperta ad una fonte dati cioè ad un Server. L'oggetto Command è un oggetto che può essere usato per eseguire una query o una Stored Procedure. L'oggetto Parameters rappresenta l'insieme dei parametri (insieme di oggetti Parameter) dell'oggetto Command. Un Parameter è un oggetto che rappresenta un parametro di un Command basato su una Stored Procedure.

Provider SQL Native, direttamente al file di database (nominati dvd.MDF).
Facciamo notare che in entrambi i casi, il Data Source è specificato nella forma nome del server SQL Express\sqlexpress.

CREARE E RIUTILIZZARE I DATABASE

Tra i tool disponibili per SQL Express ci sono anche due amministratori di database che permettono di creare e manipolare i principali elementi di SQL Server. I tool in questione sono Express Manager e SQL Server Management Studio, come accennato, entrambi scaricabili dal sito delle Microsoft. SQL Server Management Studio è un tool integrato in MS Visual Studio 2005 ma utilizzabile anche dall'esterno. Esso include le funzionalità di SQL Server 2000 Enterprice Manager, di Analysis Manager e Query Analyzer e tra l'altro permette di scrivere MDX, XMLA e XML Statements. Express Manager, invece, è ridotto nelle funzionalità, però è utile per creare database ed eseguire query. Un altro modo, poco ortodosso, per creare database nel nuovo ambiente è quello di riutilizzare i database di MSDE o SQL Server. In altre parole, basta copiare il file che contiene il database, cioè i file .MDF, e connetterlo all'applicazione, e quindi ad SQL Server, come file esterno. Di seguito spieghiamo come connettere un'applicazione Visual Basic 6 ad un database da riutilizzare (o nuovo).

1. Copiate un file MDF nella cartella C:/data;

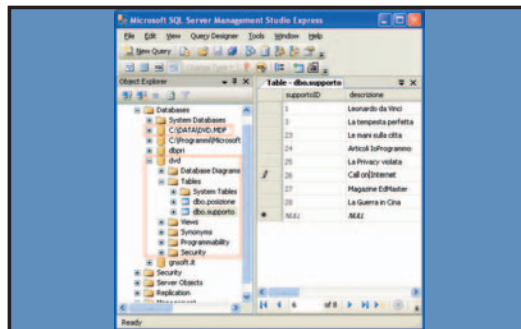



Fig. 2: ***L'interfaccia di SQL Server Management Studio Express.***

2. Create un nuovo progetto Visual Basic e referenziate la libreria MS Activex Data Objects 2.5 (cioè ADODB);
3. Sulla Form1 disponete un pulsante, nel quale inserirete il codice per collegarsi al database ed eseguire una query. In partico lare la query descritta sotto permette di ricavare il numero di record di una tabella.

```
Private Sub Command1_Click()
    Set connectionADO = New ADODB.Connection
    Set rstdvd = New ADODB.Recordset
    With connectionADO
     .ConnectionTimeout = 30 'secondi
     .CommandTimeout = 200
     .ConnectionString =
        "Provider=SQLNCLI.1;Integrated Security=SSPI;" & _
     "AttachDBFileName=" & "C:\Data" &
                                   "\nomevostroDB.MDF;" & _
     "Data Source=nomevostroserver\sqlexpress"
     .Open
    End With
    rstdvd.Open "Select * from nometabella",
             connectionADO, adopenkeyset, adlockoptimistic
    If Not rstdvd.EOF Then
     MsgBox "Numero record nella tabella: " +
                                 CStr(rstdvd.RecordCount)
    End If
End Sub
```

Nella procedura precedente dopo la connessione al database, viene eseguita una query Select su una tabella (nometabella). Se questa tabella contiene almeno un record viene mostrato un MsgBox, con il numero di record cioè, con il valore della proprietà RecordCount.

ARCHIVIARE CD E DVD

L'applicazione di esempio permette di creare un archivio con una serie di informazioni sui dati contenuti in dei supporti multimediali (DVD, CD ecc.) e sul posto dove sono riposti. Per esempio sarà possibile sapere se in un dato DVD, posto in un certo contenitore, sono archiviati film, piuttosto che canzoni, piuttosto che programmi … . Inoltre è possibile inserire informazioni sugli attori, sui cantanti ecc. Ma la cosa più interessante, anche dal punto di vista della programmazione, è che l'applicazione permette di gestire le immagini delle copertine dei supporti ottici. La gestione delle immagini, in generale, può essere fatta in due modi: salvando nel database soltanto il Path dell'immagine e predisponendo una directory, di supporto, che conterrà tutte le immagini; oppure, salvando direttamente le immagini in un campo di tipo image del database. Noi presenteremo soltanto la seconda modalità. Il database dell'applicazione presenta diverse tabelle, in questo appuntamento introduciamo soltanto le tabelle Supporto e Posizione. La Supporto contiene le principali informazioni dei supporti ottici, Posizione, invece, contiene informazioni sui posti in cui sono conservati i supporti ottici (armadi, cassetti, contenitori …). I campi delle tabelle sono riassunti nelle Tabella 2. Facciamo notare che la Key della tabella Supporto è supportoID, che è un tipo di campo che crea una sequenza numerica per identificare i nuovi record, questo grazie alla proprietà IDENTITY(1,1).

Le due tabelle sono in relazione attraverso il campo supportoID. Notate, inoltre, che le immagini sono salvate nel campo CopertinaImm di tipo image. Nel CD allegato alla rivista troverete gli Script per la creazione del database e delle tabelle introdotte.

	Supporto
Campo	**Tipo**
supportoID	int IDENTITY(1,1) NOT NULL
Descrizione	Nvarchar (250) NULL
Tipo	Nvarchar (50) NULL
Durata	Nvarchar (50) NULL
Critica	ntext NULL
Data	Datetime NULL
Masterizzato	Nvarchar (2) NULL
Copertinaimm	Image NULL
	Posizione
Campo	**Tipo**
supportoID	int NOT NULL
Descrizione	Nvarchar (100) NULL

Tabella 2: *Due tabelle del database DVD*

ADO STREAM E CAMPO IMAGE

In questo paragrafo presentiamo uno stralcio dell'applicazione che gestisce la mediateca. In particolare spieghiamo come gestire le immagini archiviate nel campo image della tabella Supporto. Questo ci ritornerà utile nel successivo appuntamento, quando presenteremo le altre parti dell'applicazione. Per manipolare i dati binari, che costituiscono un'immagine, conviene utilizzare l'oggetto ADO Stream. Questo è presente in ADO dalla versione 2.5. ADO Stream. In parole semplici, è un file salvato in memoria, e non sul disco, che riproduce uno Stream (sequenza) di dati binari o testo. Lo Stream può essere ottenuto da un Record o da un URL (file). ADO Stream presenta vari metodi tra i quali: Open per attivare l'oggetto; Close per disattivarlo; Write e WriteText per scrivere Stream di Byte o testo;Read e ReadText per leggere Byte o testo dallo Stream; SaveToFile e LoadFromFile salva o ripristina il contenuto dello Stream in/da un file esterno. Infine è presente la proprietà Type che determina il tipo di dato gestito dallo Stream (adTypeBinary, adTypeText).

Di seguito per punti presentiamo un progetto Visual Basic che consente di leggere, salvare e modificare il contenuto del campo image della tabella Supporto.

1. Dopo avere creato il database DVD con la tabella Supporto, create un nuovo progetto Visual Basic che referenzia le librerie MS Activex Data Objects 2.5, e i componenti MS Common Dialog Control 6.0 e MS ADO Data Control 6.0.

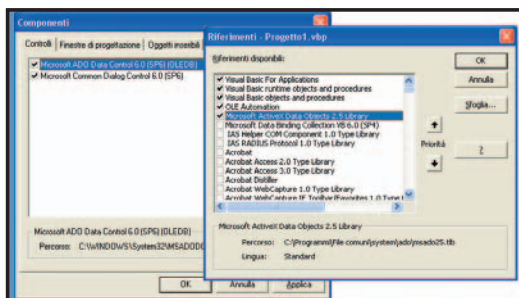



Fig. 2: *I riferimenti del progetto.*

2. Su una Form (nominata FrmImages) predisponete, tra l'altro, un Common Dialog e un oggetto ADODC. Il Common Dialog serve per ricercare un file immagine, l'ADODC serve per scorrere i Record della tabella. Gli altri elementi da inserire sulla form sono due TextBox, uno per la descrizione (TxtTitolo) del supporto ottico e uno per il codice (TxtSupportoID). Un PictureBox (Picture) per l'immagine contenuta nel campo CopertimaImm del database. Quattro Command Button da utilizzare rispettivamente per caricare un record (pulsante nominato CaricadaDB), per cambiare (pulsante Cambiaimm) l'immagine mostrata nella Picture1, per cancellare (CancelladaDB) l'immagine dalla Picture1 e dal database e per mostrare l'immagine (pulsante Allarga) su una seconda Form che contiene un sistema di scorrimento delle immagini. Le dichiarazioni globali da inserire nella FrmImages sono tre oggetti ADO cioè una Connessione, un Recordset e uno Stream.

```
Dim ConnectionADO As ADODB.Connection
Dim Rstdvd As ADODB.Recordset
Dim ImmStream As ADODB.Stream
```

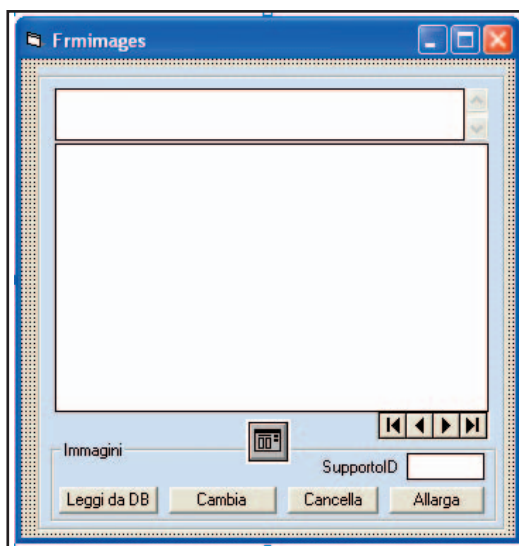
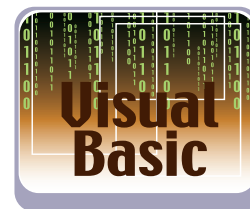
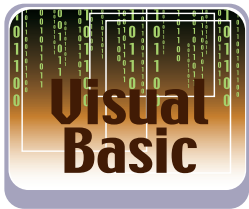



Fig. 4: *La form in progettazione.*

ODBC E OLEDB

La tecnologia ODBC (Open Database Connectivity) serve per interfacciare le applicazioni ai database relazionali. Essa è basata su Structured Query Language (SQL). La tecnologia ADO/OLEDB (COM-Based), invece, permette di accedere a qualsiasi fonte dati (relazionali e non). In tale contesto, OLE DB (un gruppo di interfacce COM) è l'interfaccia di basso livello, mentre ADO (un insieme di oggetti di alto livello) è l'interfaccia di programmazione. Le tecnologie ODBC e OLEDB sono integrate attraverso il Provider OLE DB per ODBC (MSDASQL).

3. La connessione al database DVD viene effettuata in due modi: con il controllo ADODC e con una procedura nominata Init, mostrata di seguito. In particolare il controllo ADODC è connesso utilizzando il DSN nominato DSNdvd che, utilizzando Native SQL Client, si collega al file c:/data/dvd.mdf. Ricordiamo che per utilizzare ADODC, bisogna predisporre il Binding dei TextBox e delle Picture (questo però, quando si modifica il campo TxtSupportoID, genera un Warning !?). La Init è la seguente

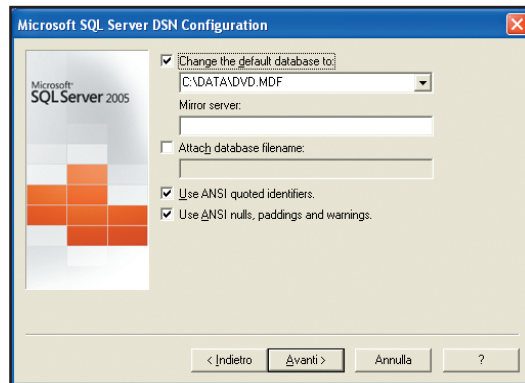



Fig. 5: **La pagina del Wizard che permette di attaccare il file MDF.**

```
Sub Init(codice As String)
 Set ConnectionADO = New ADODB.Connection
 Set Rstdvd = New ADODB.Recordset
 Dim wherecodice As String
 If codice <> "" Then
  wherecodice = " where supportoid = " + codice
 End If
 With ConnectionADO
  .ConnectionTimeout = 30
  .CommandTimeout = 100
  .ConnectionString =
        "Provider=SQLNCLI.1;Integrated Security=SSPI;" _
    & "AttachDBFileName=" & "C:\Data" _
    & "\dvd.MDF;Data
                        Source=vostroserver\sqlexpress"
  .Open
  End With
 Rstdvd.Open "Select * from supporto " +
                                      wherecodice, _
 ConnectionADO, adOpenKeyset, adLockOptimistic
 If Rstdvd.EOF Then
   Set Rstdvd = Nothing
 End If
End Sub
```

La Init ha come parametro l'identificatore del Record cioè il valore di TxtSupportoID. La stringa di connessione usa SQLNCLI.1 per connettersi al file c:/data/dvd.mdf. Il RecordSet Rstdvd è caricato con i valori di un record della tabella Supporto (se esiste).

4. Per ricercare e mostrare i dati di un record specifico si può procedere in due modi: premendo il pulsante Invio, quando il cursore è dentro il TxtSupportoID, oppure cliccando il pulsante CaricadaDB. Il codice per queste azioni è il seguente.

```
Private Sub txtsupportoID_KeyPress
                                   (KeyAscii As Integer)
On Error Resume Next
If KeyAscii = Asc(vbCr) Then
If Me.txtsupportoID <> "" Then
    CaricadaDB_Click
End If
End If

Private Sub CaricadaDB_Click()
 If Me.txtsupportoID <> "" Then
  Init (txtsupportoID)
 Else
  MsgBox "Specificare un codice"
  Exit Sub
 End If
 If Rstdvd Is Nothing Then
  MsgBox "Record non trovato"
  Picture1.picture = Nothing
  Exit Sub
 End If
 If IsNull(Rstdvd.Fields("copertinaimm").Value) Then
  MsgBox "Non trovata immagine"
  Picture1.picture = Nothing
  Exit Sub
 End If
  Set ImmStream = New ADODB.Stream
  ImmStream.Type = adTypeBinary
  ImmStream.Open
  ImmStream.Write
                      Rstdvd.Fields("copertinaimm").Value
  ImmStream.SaveToFile "C:\Temp.bmp",
                                   adSaveCreateOverWrite
  Picture1.picture = LoadPicture("C:\Temp.bmp")
  Kill ("C:\Temp.bmp")
  ConnectionADO.Close
End Sub
```

Notate che per caricare l'immagine dal database siamo costretti ad utilizzare un oggetto Stream e un file immagine di comodo, nominato Temp.bmp, che prima di chiudere la procedura viene eliminato con Kill. Notate anche che i dati dell'immagine (campo copertinaimm) sono prima copiati nello Stream, poi nel file esterno di supporto ed infine sono caricati nel controllo Picture1, il passaggio diretto è praticamente impossibile.

5. Per caricare un'immagine esterna nel database utilizziamo la procedura Cambiaimm_Click

```
Private Sub Cambiaimm_Click()
 If Me.txtsupportoID = "" Then
  MsgBox "Specificare un codice"
  Exit Sub
 End If
 With dlgDialog
  .DialogTitle = "Apri Immagine"
  .Filter = "Image Files (*.gif; *.bmp;*.jpg)|" _
  & "*.gif;*.bmp;*.jpg"
  .ShowOpen
  If .FileName = "" Then
   MsgBox "File non trovato o non valido", _
   vbOKOnly + vbExclamation, "Attenzione"
  Else
   Salva .FileName
  End If
 End With
 Me.Adodc1.Refresh
 'riporta al primo record
End Sub
```

Nella Cambiaimm_Click l'istruzione Adodc1.Refresh esegue la query del controllo ADODC (ed imposta il suo cursore sul primo record). Per salvare l'immagine nel database è invocata la procedura Salva con il nome del file immagine.

```
Private Sub Salva(Nomefile As String)
 init txtsupportoID
 If Rstdvd Is Nothing Then
  MsgBox "Il record non esiste," _
  & "non è possibile inserire immagini"
  Exit Sub
 End If
 Dim TempPic As StdPicture
 Set TempPic = LoadPicture(Nomefile)
 If TempPic Is Nothing Then
  MsgBox "Immagine non valida", vbOKOnly,
                                   "Attenzione"
  Exit Sub
 End If
  Set ImmStream = New ADODB.Stream
  ImmStream.Type = adTypeBinary
  ImmStream.Open
  ImmStream.LoadFromFile Nomefile
  Rstdvd.Fields("copertinaimm") = ImmStream.Read
  Rstdvd.Update
  Picture1.picture = LoadPicture(Nomefile)
  ImmStream.Close
  ConnectionADO.Close
End Sub
```

Notate che per salvare un'immagine nel database si utilizza l'oggetto Stream caricato con un file immagine esterno. Inoltre, notate, che per verificare se l'immagine esiste, ed è valida, utilizziamo un oggetto StdPicture, questo perché non è possibile creare un oggetto Picture con del codice come il seguente Dim Imm As New Picture.

Fig. 6: ***La Form in fase di esecuzione.***

6. Per cancellare un'immagine dal database e dalla Picture1 utilizziamo la seguente.

```
Private Sub CancelladaDB_Click()
 Init txtsupportoID
 Rstdvd.Fields("copertinaimm").Value = Null
 Rstdvd.Update
 ConnectionADO.Close
 Picture1.picture = Nothing
 Adodc1.Refresh
End Sub
```

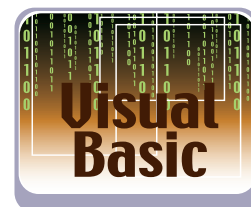
7. Infine per mostrare l'immagine su una Form (FrmVista) che contiene un sistema di scorrimento, utilizziamo la seguente procedura.

```
Private Sub Allarga_Click()
 Frmvista.caricaimmagine Me.Picture1
End Sub
```

Caricaimmagine è una procedura pubblica della FrmVista che imposta l'immagina contenuta nella Picture1 nel sistema di scrolling. La FrmVista è presentata tra i Tips di questo mese e la trovate nel CD allegato alla rivista. Tra i Tips è presentato anche un metodo che permette di mostrare i dati della tabella Supporto in una griglia.

CONCLUSIONE

Nel prossimo appuntamento amplieremo le conoscenze su SQL Server 2005, descriveremo come utilizzare le Stored Procedure e presenteremo le parti mancanti dell'applicazione.

Massimo Autiero

AMMINISTRAZIONE ODBC E DSN

L'Amministratore Origini dati ODBC permette di creare i DSN (Data Source Name). Ad esso si accede attraverso il menu strumenti di amministrazione del sistema Operativo o dal Pannello di controllo. Le connessioni alle fonti dati (per esempio Access o SQL Server) possono essere fatte se si è in possesso dei driver necessari (si controlli la scheda DSN utente dell'amministratore). Nel caso di SQL Express deve esserci il Driver SQL Native Client. Per creare un DSN basta seguire gli Step del Wizard Aggiungi DSN Utente questo permette di scegliere il Driver, il tipo e il nome del database. Per i nostri esempi abbiamo creato il DSN Utente nominato DSNdvd connesso al file C:\data\dvd.mdf.

JAVA GESTISCE LA SICUREZZA

NELL'INTRICATO MODO DELLA SICUREZZA, JAVA USA UN PROPRIO STANDARD FACILE E POTENTE AL TEMPO STESSO. REALIZZIAMO INSIEME UN'APPLICAZIONE CHE NE SPIEGA PRINCIPI E FUNZIONALITÀ.

Il problema è molto comune: disponiamo di un sistema Unix e vogliamo che una volta lanciata l'applicazione l'utente debba inserire le proprie credenziali di autenticazione prima di poterla usare. Apparentemente l'idea è semplice, basta confrontare i dati immessi con quelli contenuti nel file di passwd di Unix. Che succede se improvvisamente vogliamo cambiare il metodo di autenticazione? Ad esempio l'applicazione potrebbe essere una Web Application, e i dati di login potrebbero essere contenuti in un database, oppure il sistema di riferimento potrebbe essere Windows. Dovremmo riscrivere interamente il codice per la gestione dell'autenticazione.
Viceversa Java mette a disposizione dei programmatori un'API studiata per porsi come framework di autenticazione, tale API prende il nome di JAAS. Richiamando i metodi di Jaas invece di scrivere un proprio codice di gestione si è sicuri dell'estrema portabilità dell'applicazione, si tratta infatti di un API "pure Java", per cui la tecnologia è estremamente portabile.

CHIARIAMOCI LE IDEE

Prima di ogni cosa Jaas deve:

1) Fornire un metodo per autenticare l'utente
2) Consentire di autenticarsi una sola volta e mantenere i propri dati di login in sessioni
3) Essere indipendente dalla tecnologia utilizzata: Kerberos, Windows NT, JNDI, Unix
4) Fornire un'interfaccia verso vari metodi di autenticazione: Smart Card, applicazione standalone, web...
5) Gestire i ruoli, ovvero concedere più o meno potere ad un utente autenticato a seconda di un ruolo ad esso attribuito all'interno di una qualunque struttura dati.

Tutto questo per quanto riguarda Jaas può essere racchiuso in una struttura modulare. Da un lato c'è il vero programma Java, dall'altro un file di configurazione che contiene almeno le informazioni relative alla tecnologia di autenticazione da utilizzare. Il primo effetto di questo tipo di struttura è quello che è sufficiente modificare una riga nel file di configurazione per variare anche il metodo di autenticazione. Per riassumere brevemente il più piccolo programma Java basato su Jaas potrebbe essere il seguente:

```
package ioprogrammo.javajaas;
import java.io.IOException;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.
                                        CallbackHandler;
import javax.security.auth.callback.
                          UnsupportedCalbackException;
import javax.security.auth.login.*;
class MyCallBackHandler implements
                                        CallbackHandler {
        public MyCallBackHandler() {
        }
        public void handle(Callback[] callbacks)
                trows IOException, UnsupportedCallback
                                        Exception {
        }
}
public class Esempio {
        public static void main(String[] args)
                                throws LoginException {
LoginContext lc=new LoginContext("ioProgrammo",
new MyCallBackHandler()); lc.login();
}}
```

Mentre il file di configurazione jaas.config conterrebbe:

```
ioProgrammo {
        com.sun.security.auth.module.
                UnixLoginModule required debug=true;
};
```

Per fare funzionare il tutto dovreste anche passare come parametro il file di configurazione alla JVM con qualcosa del genere:

```
java -Djava.security.auth.login.config==.
                    /ioprogrammo/javajaas/jaas.config
                        ioprogrammo.javajaas.Esempio
```

COME FUNZIONA?

La chiave di tutto sta nella Classe LoginContext e nel metodo login. Una volta instanziato un LoginContext esso "dialoga" con il sistema sottostante tramite un "LoginModule" che viene tipicamente definito nel file di configurazione. In sostanza i LoginModule derivano da javax.security.auth.spi.LoginModule e portano in dote tutto il codice di backend che viene richiamato dal metodo login per gestire l'autenticazione. I LoginModule prendono i dati in input tramite un Handler. Scrivere l'handler significa ad esempio consentire all'utente di immettere i dati tramite console, web, smartcard o altro.

UNA DEFINIZIONE PIÙ RIGOROSA

Le classi e le interfacce di JAAS possono essere divise in 3 gruppi:

1) generali (Subject, Principal, Credential),
2) autenticazione (LoginContext, LoginModule, CallBackHandler, CallBack),
3) autorizzazione. La classe chiave è Subject che rappresenta colui che effettua una richiesta e può essere sia una persona sia un altro programma. Una volta che il Subject è autenticato, viene popolato dai Principal, le varie identità ad esso associate. Esempi di identità sono il nome o il codice fiscale. Un Subject può anche avere associato degli attributi relativi alla sicurezza, denominati Credential, come chiavi private e certificati pubblici.

Come già accennato all'inizio, il nostro programma effettua l'autenticazione di un utente che inserisce le sue credenziali, username e password nel modulo di login. Per effettuare l'autenticazione di un Subject, l'applicazione segue una procedura precisa, abbastanza semplice da seguire, quasi tutto il meccanismo è pilotato dal LoginContext:

1) Instanziazione del LoginContext, il contesto specifico dell'applicazione
2) il LoginContext verifica la configurazione per caricare i LoginModule, i moduli dove inserire username e password
3) il LoginContext invoca il metodo login
4) Login invoca tutti i LoginModule e ognuno di essi tenta di autenticare il Subject
5) Il LoginContext restituisce il risultato dell'autenticazione
6) In caso positivo, l'applicazione restituisce il Subject al LoginContext.

LE IDENTITÀ

Lanciando il piccolo esempio di cui sopra, vi accorgerete che otterrete in output qualcosa del genere:

```
[UnixLoginModule]: succeeded importing info:
            uid = 1001
            gid = 1001
            supp gid = 4
            supp gid = 20
            supp gid = 21
            supp gid = 24
            supp gid = 25
            supp gid = 26
            supp gid = 29
            supp gid = 30
            supp gid = 44
            supp gid = 46
            supp gid = 104
            supp gid = 105
            supp gid = 106
            supp gid = 1001
        [UnixLoginModule]: added UnixPrincipal,
                UnixNumericUserPrincipal,
                UnixNumericGroupPrincipal(s),
            to Subject
```

Il punto è che stiamo usando un modulo di autenticazione Unix. Il che significa che per lanciare il programma dobbiamo in qualche modo già esserci loggati al sistema e perciò autenticati. Il parametro Required Debug=true non fa altro che stampare a video le informazioni relative all'utente che in quel momento sta usano il programma. Ovvero LoginContext dispone di un Subject che gli è stato restituito dalla corretta autenticazione dell'utente. Quello che dobbiamo fare ora è ottenere gli oggetti Principal il cui insieme costituisce il Subject dell'utente. Nel caso di un modulo Unix otterremo il nome di login dell'utente, il suo gruppo di lavoro e altre informazioni particolarmente utili. Il metodo che ci restituisce i principal è GetPrincipals che ritorna un oggetto Java.util.set che è direttamente stampabile. Un esempio d'uso è il seguente:

```
public static void main(String[] args)
                        throws LoginException {
        LoginContext loginContext=new
                    LoginContext("ioProgrammo",
                    new MyCallBackHandler());
                    loginContext.login();
        Subject subject = loginContext.getSubject();
        Set principals = subject.getPrincipals();
        System.out.println (principals);
```

```
}
```

che restituisce qualcosa del genere

```
[UnixPrincipal: fabio,
UnixNumericUserPrincipal: 1001,
UnixNumericGroupPrincipal [gruppo primario]:
....
```

MODULI PERSONALIZZATI

Avendo ben chiaro il meccanismo interno delle Jaas, possiamo procedere a scrivere un nostro modulo di autenticazione. Non saremo più supportati da Unix. Il che significa che nel file jaas.config dovrà essere contenuta una stringa che indica il modulo di autenticazione che vogliamo usare, nel nostro caso:

```
IoProgrammo {
  net.ioprogrammo.javajaas.IoProgrammoLogin
                                  Module required debug=true;
};
```

Il nostro modulo non farà molto. Semplicemente riconoscerà solo l'utente ioProgrammo con Password "ioProgrammoPassword". Lo scopo è evidentemente didattico, potete poi scrivere i vostri moduli personalizzando quello base.
Di fatto un modulo di gestione deve implementare un metodo "Initialize" che inizializzerà gli oggetti principals che compongono il subject, un metodo login che restituirà un booleano che indica se l'autenticazione è andata a buon fine o meno, un metodo commit che in caso di esito positivo dell'autenticazione associa i vari principal all'utente, un metodo abort che termina il processo di autenticazione e infine un metodo logout richiamato alla disconnessione dell'utente.
Iniziamo dal metodo "inizialize"

```
public class IoProgrammoLoginModule implements
                                        LoginModule {
   private Subject subject;
   private CallbackHandler callbackHandler;
   private Map sharedState;
   private Map options;
   private boolean debug = false;
   private boolean succeeded = false;
   private boolean commitSucceeded = false;
   private String username;
   private char[] password;
    private IoProgrammoPrincipal userPrincipal;
    public void initialize(Subject subject,
                              CallbackHandler callbackHandler,
                       Map sharedState, Map options) {
     this.subject = subject;
     this.callbackHandler = callbackHandler;
     this.sharedState = sharedState;
     this.options = options;
     // inizializzazione delle opzioni
     debug = "true".equalsIgnoreCase((String)
                                  options.get("debug"));
   }
```

Si nota subito che il metodo initialize riceve in input un Subject da valorizzare e l'handler da utilizzare per ottenere i dati dell'utente.
Il metodo login invece potrebbe essere il seguente:

```
  public boolean login() throws LoginException {
    // user name e password
    if (callbackHandler == null)
              throw new LoginException("Error: nessun
                     CallbackHandler disponibile " +
                     "per gestire l' autenticazione");
              Callback[] callbacks = new Callback[2];
      callbacks[0] = new NameCallback ("user name: ");
                  callbacks[1] = new PasswordCallback
                                ("password:", false);
    try {
      callbackHandler.handle(callbacks);
      username = ((NameCallback)
                         callbacks[0]).getName();
      char[] tmpPassword = ((PasswordCallback)
                         callbacks[1]).getPassword();
      if (tmpPassword == null) {
        // password NULL è una password vuota
        tmpPassword = new char[0];
      }
      password = new char[tmpPassword.length];
      System.arraycopy(tmpPassword, 0,
          password, 0, tmpPassword.length);
      ((PasswordCallback) callbacks[1]).clear
                                       Password();
    } catch (java.io.IOException ioe) {
      throw new LoginException(ioe.toString());
    } catch (UnsupportedCallbackException uce){
      throw new LoginException
             ("Error: " + uce.getCallback().toString() +
          " impossibile a gestire le informazioni
                                    dall' utente ");
    }
    // print debugging information
    if (debug) {
      System.out.println("\t\t[LoginModule] " +
          "user name inserita: " +
          username);
      System.out.print("\t\t[LoginModule] " +
          "password inserita: ");
      for (int i = 0; i < password.length; i++)
        System.out.print(password[i]);
      System.out.println();
    }
    // varifica username/password
    boolean usernameCorrect = false;
    boolean passwordCorrect = false;
```

APPROFONDIMENTI

SINGLE SIGN-ON

Il Single Sign-On è l'autenticazione unica, da parte di un client, per accedere a tutti i servizi offerti su server che usano sistemi operativi differenti. Senza questo sistema, l'utente si deve autenticare su ogni server.

```
    if (username.equals("IoProgrammo")) {
        usernameCorrect = true;
    }
    if (usernameCorrect &&
            password.length == 19 &&
            password[0] == 'I' &&
            password[1] == 'o' &&
            password[2] == 'P' &&
[...]
            password[17] == 'r' &&
            password[18] == 'd') {
        passwordCorrect = true;
        if (debug)
            System.out.println ("\t\t[LoginModule] "+
                "autenticazione riuscita");
        succeeded = true;
        return true;
    } else {
        // autenticazione fallita
        if (debug)
            System.out.println
                ("\t\t[LoginModule] " +
                "autenticazione fallita");
        succeeded = false;
        username = null;
        for (int i = 0; i < password.length; i++)
            password[i] = ' ';
        password = null;
        if (!usernameCorrect) {
            throw new FailedLoginException
                ("User Name sbagliato");
        } else {
            throw new FailedLoginException
                ("Password sbagliata ");
        } }}
```

Si noti l'inizializzazione di due oggetti CallBack

```
callbacks[0] = new NameCallback ("user name: ");
callbacks[1] = new PasswordCallback
                ("password: ", false);
```

Che fanno parte di un array. Questi due oggetti saranno passati più tardi in input al gestore dell'handler di cui abbiamo sommariamente parlato.
Non ci resta appunto che gestire l'Handler.
Si noti che l'handler ha come parametro un array che deve essere utilizzato proprio per riempire i dati che abbiamo visto nel metodo di login. Nel nostro caso assomiglia a qualcosa del genere:

```
public void handle(Callback[] callbacks)
        throws IOException, UnsupportedCallback
                Exception {
    for (int i = 0; i < callbacks.length; i++) {
        if (callbacks[i] instanceof NameCallback){
            NameCallback nameCallback =
                (NameCallback) callbacks[i];
            System.err.print(nameCallback.getPrompt());
            System.err.flush();
            nameCallback.setName((new Buffered
                Reader
                (newInputStreamReader
                (System.in))).readLine());
        } else if (callbacks[i] instanceof
                PasswordCallback) {
            PasswordCallback passwordCallback =
                (PasswordCallback) callbacks[i];
            System.err.print(passwordCallback.
                getPrompt());
            System.err.flush();
            passwordCallback.setPassword
                (readPassword(System.in));
        } else {
            throw new UnsupportedCallbackExcetion
            (callbacks[i], "Callback non riconosciuto");
        } } }
```

Al di la dei problemi di sintassi e in parte anche di semantica, se è ben chiaro il funzionamento di Jaas possiamo fermarci qui. Nell'esempio allegato alla rivista trovate l'intero codice, seguendo articolo e codice intero vi risulterà abbastanza chiaro l'intero meccanismo di Jaas.

CONCLUSIONI

JAAS è una libreria PAM, Pluggable Authentication Module e che, quindi, è indipendente dall'applicazione sottostante. La sua forza sta nella capacità di poter cambiare facilmente sistema di autenticazione riscrivendo i moduli e nella possibilità di definire dei Subject personalizzati.

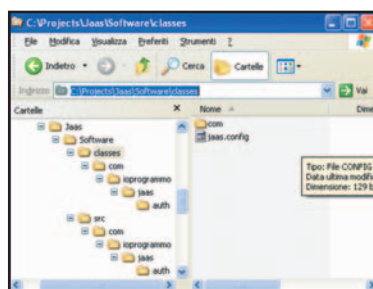
Cristiano Bellucci

GLOSSARIO

SUBJECT

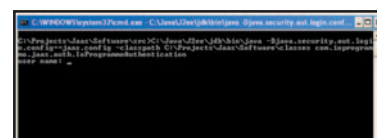
Java definisce il termine Subject come il rappresentante di una richiesta. Può essere qualsiasi entità, come una persona o un servizio ed è popolato da identità o Pricipal. Il package di riferimento è javax.security.auth.

PRIMI PASSI

LA STRUTTURA

1 Creiamo una struttura di directory per il progetto mettendo il file jaas.config nella cartella \classes e i file sorgenti (.java) in <directory_progetto>\src\com\iopro grammo\jaas\auth.

COMPILAZIONE

2 Compiliamo il codice, indicando dove sono i sorgenti e dove vogliamo mettere i file compilati.

ESECUZIONE

3 Lanciamo l'applicazione. Se tutto è andato bene apparirà la scritta "user name:"

UN PONTE TRA XML E CLASSI JAVA

INTRODUZIONE A JAXB, IL FRAMEWORK INTEGRATO NEL PACCHETTO JAVA WEB SERVICES, CHE CONSENTE DI CREARE E MANIPOLARE DOCUMENTI XML ATTRAVERSO CLASSI JAVA. VEDIAMO COME UTILIZZARLO IN MODO FACILE ED EFFICIENTE

Il meta-linguaggio XML *(Extensible Markup Language)* è un diretto discendente dell'ormai datato SGML *(Standard Generalized Markup Language)*, nato intorno agli Anni 70 per essere usato principalmente da alcuni enti governativi americani, nei processi di creazione e pubblicazione di manuali tecnici. Sebbene il consorzio ISO *(International Organization for Standardization)* lo abbia "elevato al grado" di standard internazionale, SGML non ha mai avuto una diffusione considerevole a causa della sua notevole complessità: i costi erano troppo elevati per pensare di abbracciare una tecnologia sicuramente potente e flessibile ma allo stesso tempo di difficile implementazione. Questi aspetti, insieme all'avvento del World Wide Web, indussero gli addetti ai lavori, capeggiati dal consorzio W3C, alla realizzazione di un nuovo meta-linguaggio, con caratteristiche funzionali equivalenti a quelle di SGML e sintassi notevolmente semplificata: nacque così XML. Oggi lo standard XML viene utilizzato su molteplici fronti come protocolli internet e applicazioni, in quanto ha la caratteristica di processare e trasferire i dati in maniera semplice ed efficiente. Nel panorama informatico attuale un'applicazione che non fa uso di XML non viene considerata di livello enterprise.
Una delle caratteristiche che ha fatto la fortuna di XML è quindi la portabilità dei dati. Il termine "portabilità" è molto caro alla comunità di sviluppatori Java, in quanto l'utilizzo di questo linguaggio permette la creazione di applicazioni indipendenti dalla piattaforma, in gergo "portabili".
Questo termine non è però l'unico elemento che accomuna le due tecnplogie: Java è nato per la realizzazione di applicazioni in rete mentre XML rappresenta uno strumento semplice e flessibile per la comunicazione di dati. L'accoppiata tra queste due realtà non poteva non risultare vincente. Nel prosieguo dell'articolo analizzeremo le varie modalità di gestione di documenti XML attraverso l'uso di Java, focalizzandoci in maniera particolare sulle tecniche di data binding, ovvero la capacità di trasformare schemi XML in classi Java.

METODOLOGIE DI GESTIONE XML

Da un punto di vista applicativo è possibile creare e manipolare documenti XML attraverso l'uso di metodologie diverse. Qualunque tecnica si basa sull'esistenza di un parser. Un parser è uno strumento che si colloca tra il documento XML e l'applicazione. Essendo a conoscenza della sintassi del meta-linguaggio è in grado di leggere ed analizzare i documenti formattati secondo tale standard. I due approcci più conosciuti sono SAX *(Simple API for XML Parsing)* e DOM *(Document Object Model)*. Entrambe le tecnologie sono incluse nel progetto JAXP *(Java API for XML Parsing)* che è parte integrante sia del pacchetto *Java Web Services Developer Pack* (JWSDP) sia della versione J2SE a partire dalla release 1.4. SAX è caratterizzato da un parsing di tipo *event-based:* il parser "naviga" all'interno del documento e notifica dei particolari eventi ad eventuali handler in ascolto. DOM, dal canto suo, presenta un approccio di tipo *object-based* (o *tree-based*): il parser costruisce in memoria una struttura dati ad albero, sostanzialmente analoga al *TreeModel* delle API *JFC/Swing*, su cui è possibile manipolare il contenuto dei dati contenuti all'interno del documento. Quest'ultimo tipo di parser, sebbene rappresenti una buona soluzione quando si trattano documenti di cui si conosce la struttura, è molto più dispendioso in termini di velocità che di utilizzo di memoria confronto a quello fornito dall'implementazione SAX il cui uso è consigliabile quando si vuole rispondere ad un determinato tipo di eventi

durante la lettura del documento.
Con le due tecniche appena descritte e l'ausilio di un linguaggio di programmazione ad oggetti, come Java, è possibile implementare dei meccanismi di data binding, ovvero creare una corrispondenza diretta tra documenti XML ed oggetti Java. Realizzare "from scratch" una tecnica con tali caratteristiche richiederebbe un notevole sforzo sia in termini di tempo sia di conoscenze. Nel prosieguo dell'articolo vedremo come JAXB ci viene incontro offrendoci interessanti funzionalità.

XML DATA BINDING CON JAXB 2.0

JAXB, che è l'acronimo di *Java Architecture for XML Binding*, è un progetto di Sun Microsystems che è a sua volta inglobato nei progetti *GlassFish* (un application server open source basato sulla piattaforma J2EE 5) e *JWSDP*. La versione 1.0, rilasciata nel Marzo 2003, nasce come risposta alla *Java Specification Request 31* ed è stata seguita da un gruppo di esperti di importanti società tra cui IBM, Oracle e HP. *JAXB* ha come fine primario quello di "legare" una generica struttura XML, definita da uno schema, ad una o più classi Java, preservando il livello concettuale dello schema stesso. Attraverso questa tecnica saremo in grado di generare in maniera automatica un mapping tra documenti XML ed oggetti Java. Il codice generato astrae, tramite delle interfacce, la gestione dello specifico documento XML consentendo la sua creazione e manipolazione senza bisogno di conoscere la sintassi del meta-linguaggio. In pratica, attraverso JAXB, lo sviluppatore Java si "svincola" dal noioso compito di gestione dei documenti attraverso SAX e DOM che richiedono una buona conoscenza dello standard. Allo stesso tempo questa architettura presenta caratteristiche prestazionali di parsing simili a quelle di SAX e capacità di memorizzazione dati più efficienti rispetto a quelle DOM.
In questo articolo introdurremo ed useremo la versione 2.0 EA (Early Access) che, oltre ad essere pienamente compatibile con la precedente, presenta delle migliorie in termini di performance ed introduce interessanti funzionalità come l'integrazione dell'architettura StaX (Streaming API for XML) o la possibilità di creare schemi XML partendo da classi di modello legate alle specifiche JavaBean. Tenteremo di sviluppare alcuni esempi significativi che vi aiuteranno nel corso del vostro lavoro giornaliero con XML e Java.

GENERARE CLASSI DALLO SCHEMA

Prima di iniziare ad usare JAXB dobbiamo creare uno schema che descriva il modello dati che si ha intenzione di trattare. Il seguente schema descrive la struttura dati di un ipotetico autosalone:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=
             "http://www.w3.org/2001/XMLSchema">
     <xs:element name="Autosalone" type=
                              "AutosaloneType"/>
     <xs:complexType name="AutosaloneType">
          <xs:sequence>
               <xs:element ref="automobili"
  minOccurs= "0" maxOccurs="unbounded"/>
               </xs:sequence>
          <xs:attribute name="nome" type=
                                   "xs:string"/>
     </xs:complexType>
     <xs:element name="automobili" type=
                              "AutomobileType"/>
     <xs:complexType name="AutomobileType">
  <xs:attribute name="marca" type= "xs:string"/>
  <xs:attribute name="tipo" type= "xs:string"/>
<xs:attribute name="prezzo" type="xs:double"/>
     <xs:attribute name="data" type="xs:date"/>
     </xs:complexType>
</xs:schema>
```

Viene definito un tipo *AutosaloneType* che

COME INIZIARE

1 **È indispensabile avere installato sul computer Java 2 Standard Edition SDK 5.0 o superiore. È preferibile utilizzare l'ultima versione disponibile dal sito http://java.sun.com/. Verificare che la variabile d'ambiente *JAVA_HOME* punti correttamente alla directory di installazione della JDK.**

2 **Scaricare il pacchetto *JAXB 2.0 Early Access* (9 MB circa) dalla sezione *Downloads* del sito https://jaxb.dev.java.net/ framework viene distribuito sotto forma di jar autoinstallante. Pertanto, è possibile installare il tool, indipendentemente dalla piattaforma sulla quale sarà utilizzato, copiando il jar nella directory in cui vogliamo installarlo (generalmente in *<SYSTEM_DRIVE_ ROOT>\Programmi* per OS Windows oppure /usr/java per OS Unix/Linux). Per avviare il processo di installazione si dovrà eseguire il seguente comando:**

```
java -jar JAXB_RI_20050622.jar
```

Dopo aver accettato i termini di licenza (CDDL), l'installer proseguirà il suo lavoro creando la cartella *jaxb-ri-20050622* estraendovi tutti i file necessari per il corretto funzionamento. La documentazione di JAXB è consultabile dal file *index.html* contenuto nella cartella *docs.*

3 **Impostare la variabile d'ambiente *JAXB_HOME* con il percorso assoluto della cartella generata dall'installer nello step precedente. Aggiungere alla variabile d'ambiente *PATH* il percorso *JAXB_HOME/bin.* Aggiungere alla variabile d'ambiente *CLASSPATH* il puntamento alle librerie *activation.jar, jaxb-api.jar, jaxb-impl.jar, jsr173_1.0_api.jar* contenute nella directory *JAXB_HOME/lib.***

PRINCIPALI CARATTERISTICHE DI JAXB 2.0

- Permette l'accesso ai dati senza bisogno di conoscere XML;
- Generazione di classi Java da schemi XML;
- Generazione di schemi XML da codice Java tramite annotations;
- Gestione dei dati in memoria;
- Garantisce la validità dei dati;
- Applicazioni facili da creare ed utilizzare;
- Buone prestazioni ed uso efficiente della memoria;
- Supporta i generics introdotti in Java 5;
- Conversione dei dati XML in tipi Java e viceversa;
- Possibilità di creare estensioni e personalizzazioni;
- Il codice sorgente è disponibile.

contiene due elementi: un nome ed una serie di elementi automobili. Il primo è di tipo stringa, mentre il secondo è una sequenza di elementi di tipo *AutomobileType*. Quest'ultimo è caratterizzato dagli elementi marca, tipo, prezzo e data.
A questo punto entra in gioco il *Binding Compiler*. Per processare le classi corrispondenti allo schema e generare automaticamente il codice, basta eseguire il seguente comando:

```
xjc.sh -d src -p ioProgrammo autosalone.xsd
```

Il compilatore così invocato (con estensione .bat su OS Windows), dopo aver effettuato il parsing dello schema, genererà il codice sorgente, con package ioProgrammo, nella cartella *src*. Le classi generate saranno tre: *AutosaloneType, AutomobileType* e *ObjectFactory*. Le prime due rappresentano il mapping delle strutture descritte nello schema, mentre l'ultima espone dei metodi di factory per facilitarne la creazione. Per meglio evidenziare il lavoro eseguito dal compilatore JAXB riportiamo il sorgente contenuto nel file *AutosaloneType.java* escludendo i commenti:

```
package ioProgrammo;
import java.util.ArrayList;
import java.util.List;
import javax.xml.bind.annotation.AccessType;
import  javax.xml.bind.
                                    annotation.Xml
AccessorType;
import javax.xml.bind.annotation.XmlAttribute;
import javax.xml.bind.annotation.XmlType;
import ioProgrammo.AutomobileType;
@XmlAccessorType(AccessType.FIELD)
@XmlType(name = "AutosaloneType")
public class AutosaloneType {
    protected List<AutomobileType> automobili;
    @XmlAttribute
    protected String nome;
    protected List<AutomobileType>
                              _getAutomobili() {
        if (automobile == null) {
            automobili = new ArrayList
                              <AutomobileType>();
        }
        return automobili;
    }
    public List<AutomobileType> getAutomobili() {
        return this._getAutomobili();
    }
    public String getNome() {
        return nome;
    }
    public void setNome(String value)
    {
        this.nome = value;
    }
}
```

È interessante notare l'uso della notazione *generics*, introdotta con la release 1.5 di Java, che rende il codice più robusto, in quanto ci libera da errori di casting a runtime. Inoltre, viene anche evidenziata la metodologia di conversione dei tipi da XML a Java, che risulta pressoché equivalente a quella implementata nella tecnologia JDBC, con alcune eccezioni come ad esempio la gestione delle date. In **Figura 1** sono elencate alcune conversioni dei tipi dati più comuni.

XML Type	Java Type
xs:int	int
xs:double	java.land.Double
xs:string	java.lang.String
xs:integer	java.math.BigInteger
xs:date	javax.xml.datatype.XMLGregorianCalendar

Fig. 1: **Conversioni comuni tra tipi di dati XML e Java**

Prima di addentrarci nella fase di manipolazione dei documenti XML è necessario compilare le classi generate precedentemente.

MARSHAL & UNMARSHAL

In questo paragrafo descriveremo il corretto uso delle classi appena generate. In prima battuta illustreremo le tecniche di marshalling, ovvero come salvare in formato XML un oggetto definito da una classe Java.

```
// istanzio il factory per la creazione
// degli elementi
// definiti nello schema
ObjectFactory factory = new ObjectFactory();
// creo un oggetto che descrive
// un tipo autosalone ...
AutosaloneType at =
                    factory.createAutosaloneType();
// ... e gli assegno un nome
at.setNome("ACME Car");
// creo una automobile ...
AutomobileType auto =
                    factory.createAutomobileType();
auto.setMarca("ACMECarFactory");
auto.setTipo("SPECTRA SW");
auto.setPrezzo(15000);
auto.setData(DatatypeFactory.newInstance().
                        newXMLGregorianCalendar(
                        "2005-02-24T00:00:00"));
```

```
// ... e la aggiungo all'autosalone
at.getAutomobili().add(auto);
// creo l'elemento xml contenente l'autosalone
JAXBElement<AutosaloneType> autosalone =
                        factory.createAutosalone(at);
// ottengo un riferimento al contesto JAXB ...
JAXBContext ctx = JAXBContext.newInstance(
                                      "ioProgrammo");
// ... e creo un oggetto marshaller ...
Marshaller m = ctx.createMarshaller();
m.setProperty(
      Marshaller.JAXB_FORMATTED_OUTPUT, true);
// scrivo nel file xml l'autosalone
m.marshal(autosalone,
        new FileOutputStream("ACMECar.xml"));
```

L'esecuzione di questo stralcio di codice ha come risultato la creazione del file *ACMECar.xml* contenente un autosalone con un'automobile: l'astrazione apportata da JAXB nasconde completamente la sintassi XML producendo il seguente risultato:

```
<?xml version="1.0" encoding="UTF-8"
                           standalone= "yes"?>
<Autosalone nome="ACMECar">
    <automobili data="2005-02-24T00:00:00"
marca="ACMECarFactory" prezzo="15000.0"
                           tipo="SPECTRA SW">
    </automobili>
</Autosalone>
```

L'*unmarshalling* definisce il processo inverso, ovvero il mapping del contenuto del file XML in oggetti Java. Vediamo come:

```
// creo l'unmarshaller ...
Unmarshaller um = context.createUnmarshaller();
// ... e faccio la load del file
// contenente l'autosalone
      JAXBElement<AutosaloneType>
autosaloneXML =
   (JAXBElement<AutosaloneType>)um.unmarshal(
            new FileInputStream("ACMECar.xml"));
AutosaloneType autosalone =
                      autosaloneXML.getValue();
// stampo a video le automobili presenti
List<AutomobileType> garage =
                     autosalone.getAutomobili();
System.out.println("Il garage contiene "
              + garage.size() + " automobili");
for (AutomobileType auto : garage) { // enhanced
                                    loop (Java 5)
System.out.println("Marca: " + auto.getMarca()
      + " - Tipo: " + auto.getTipo() + " - Prezzo: "
              +auto.getPrezzo() + " - Data: " +
                              auto.getData());}
```

SCHEMA GENERATOR

Gli argomenti finora trattati descrivono funzionalità che, confronto alla precedente versione del framework, hanno subito dei miglioramenti, sia in termini prestazionali sia di facilità di utilizzo. In questo paragrafo parleremo invece di una nuova feature, introdotta con la versione 2.0 di JAXB: la generazioni di schemi XML da classi Java. Per eseguire questo processo, che è l'inverso di quello illustrato nel paragrafo precedente, è necessario l'uso delle *annotations* definite nel package *javax.xml .bind.annotations*. Tale tecnica, introdotta nella J2SE 1.5, definisce dei particolari "metadati" utilizzabili in svariati modi. Il framework JAXB li adotta per acquisire informazioni riguardanti la struttura dei documenti XML mappati dalle classi Java. Inoltre, è importante sottolineare che i sorgenti da cui generare gli schemi devono essere conformi alle specifiche JavaBean. L'esecuzione dello script *schemagen* (disponibile sia per Windows che per OS Unix like), contenuto nella cartella *JAXB_HOME/bin*, seguito dal percorso della classe ne genera lo schema corrispondente.

CONCLUSIONI

In questo articolo sono state illustrate le tecniche base di XML facility fornite dal framework JAXB 2.0 che, allo stato attuale, è in fase di rilascio finale. È inoltre importante sapere che le novità introdotte da questa neonata release non si limitano solo a questo, ma integrano anche altre utili funzionalità come ad esempio la possibilità di effettuare un mapping parziale del documento XML all'interno delle classi Java e tanto altro ancora.

Fabrizio Fortino

SUL WEB

http://ws.apache.org/jaxme
http://xmlbeans.apache.org
http://jbind.sourceforge.net
http://www.relaxer.org/

BINDMARK

BindMark è l'unione delle parole binding è benchmark. Si tratta di un progetto, ospitato sul portale *java.net,* che ha come scopo il testing dei framework di XML *Data Binding* per il linguaggio Java. Attualmente le applicazioni analizzate sono 21, sia di carattere commerciale che open-source. É possibile suddividere i test in due macro aree: performance e facilità d'utilizzo. La prima area è a sua volta suddivisa in tre categorie, basate sulla dimensione dei dati trattati: *small, medium, large.* In tutte le sottocategorie JAXB 2.0 si afferma alla quinta posizione, con un trend che migliora man mano che il set di dati trattati aumenta. Per quanto riguarda la facilità di utilizzo sono stati analizzati vari fattori come il tempo necessario alla prima esecuzione o le modalità di import: per le sue caratteristiche JAXB 2.0 si attesta alla prima posizione. Per maggiori dettagli è possibile visitare il sito http://bindmark.dev.java.net/.

L'ALLENATORE DEL FANTACALCIO

IMPARIAMO AD UTILIZZARE IL FRAMEWORK SPRING ED IL PATTERN IOC PARTENDO DALLA REALIZZAZIONE DI UN SEMPLICE SISTEMA ESPERTO PER LA SELEZIONE DELLA SQUADRA DEL FANTACALCIO

La storia di Java è costellata di "framework" di vario genere. Dove per framework si intende un insieme di librerie e tool di sviluppo costruite al di sopra del linguaggio base e che ne estendono le funzionalità. Tutto ciò ha fatto sì che Java disponga di un enorme numero di librerie che assolvono pressoché ad ogni funzione, ma che tali librerie se usate l'una insieme all'altra spesso non comunichino o provochino addirittura problemi di incompatibilità. Per risolvere questo problema è nato Spring, una sorta di FrameWork dei FrameWork, che riunisce sotto un unico cappello un grande numero di librerie al fine di farle cooperare in modo significativo fra loro.
Spring è un framework che favorisce l'utilizzo e l'integrazione di tecnologie già affermate e solo in pochi casi fornisce delle nuove soluzioni. La caratteristica principale di Spring però è l'utilizzo di un pattern denominato Inversion of Control che consente di sviluppare le classi dell'applicazione con un alto grado di "disaccoppiamento". Ciò come vedremo produce notevoli vantaggi.

UN'APPLICAZIONE D'ESEMPIO

In questo articolo scriveremo un'applicazione di esempio senza tener presente l'utilizzo di Spring. Solo successivamente lo utilizzeremo e ne valuteremo i vantaggi apportati. Prima di partire con il codice dobbiamo fare un'importante considerazione: se possiamo scrivere il codice e solo in un secondo tempo introdurre Spring, significa che questo framework non è invasivo e permette un suo utilizzo anche solo parziale. L'applicazione di esempio è un sistema esperto che fornisce all'utente la squadra del Fantacalcio da schierare in base allo storico delle prestazioni effettuate dai singoli calciatori.
L'applicazione dovrà tener conto di alcune variabili:

1) Lo schema di gioco che l'allenatore della squadra vorrà adottare, potrà essere ad esempio un 4-4-2 piuttosto che un 4-3-3 etc. Ovviamente dovremo predisporre una struttura idonea a contenere e gestire queste informazioni

2) La formazione dovrà essere stilata in base a varie opzioni, ad esempio la media del voto del rendimento dei singoli, oppure schierando sempre nel ruolo il calciatore che nell'ultima partita ha avuto il voto più alto, in modo da schierare solo i calciatori più in forma. Per gestire questo genere di flessibilità avremo anche in questo caso bisogno di strutture e metodi appositi. Vedremo come Spring potrà aiutarci in questo

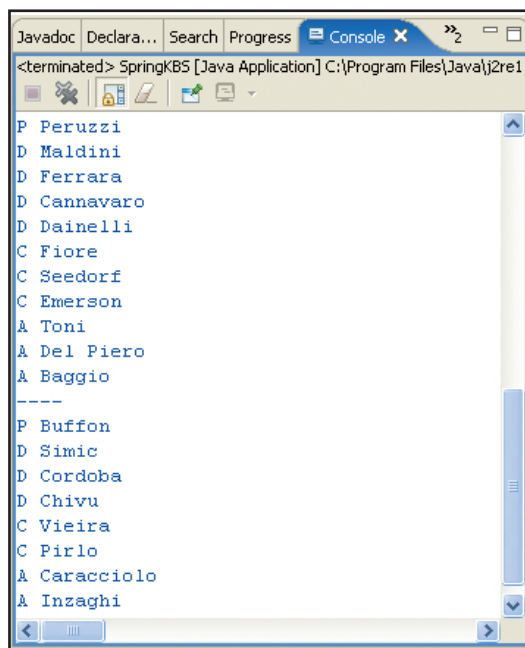



Fig. 1: L'output fornito dall'applicazione

Fatte queste considerazioni possiamo già individuare gli attori della nostra applicazione. È necessario infatti avere a disposizione la cosiddetta base della conoscenza (KB: Knowledge Base) che fornisce l'elenco dei calciatori (nome e ruolo) ed i voti da essi ottenuti. La base della conoscenza deve essere fornita ad un motore inferenziale che, in base ai vari criteri stabiliti dall'utente, stila i diciotto calciatori da proporre per la prossima partita. In sostanza ci servirà un'interfaccia che accede a un database, recupera i dati collegati ai giocatori e valorizza con questi dati una seconda interfaccia che procede a costruire la formazione.

DEFINIAMO LE INTERFACCE

La nostra interfaccia sarà la seguente.

```
public interface KBI {
    public ArrayList getTeam();
}
```

Espone il solo metodo getTeam() che fornisce parte dei dati necessari al calcolo della formazione. Analizziamo adesso l'interfaccia che definisce il comportamento della classe che si occuperà del calcolo della formazione titolare e delle riserve.

```
public interface EngineI {
    public void setSchema(String schema);
    public void setComparator(Comparator co);
    public void setKb(KBI kb);
    public ArrayList getTeam();
}
```

Il metodo setSchema(...) serve a fissare lo schema di gioco che si vuole utilizzare; a tal proposito stabiliamo già da adesso la convenzione secondo la quale lo schema è rappresentato da una stringa composta da tre digit la cui somma è 10, essi rappresentano rispettivamente i calciatori che compongono la difesa, il centrocampo e l'attacco della nostra squadra. "442", "532" e "433" sono alcuni schemi utilizzabili. Un altro metodo di cui il nostro Engine sarà dotato è setComparator(...), questo metodo serve a stabilire il criterio di ordinamento (un oggetto che implementa java.util. Comparator) con cui selezionare i componenti dei vari reparti. Inoltre il metodo setKb(...) serve a passare la base della conoscenza all'Engine. Infine il metodo getTeam(...) restituirà un ArrayList di diciotto calciatori selezionati in base allo schema e al criterio di ordinamento desiderato.

COSA SERVE PER COMINCIARE

Per utilizzare Spring è necessario scaricare il framework presso il suo sito www.springframework.org. Una volta scaricato lo zip, bisogna scompattarlo in locale; sotto la directory dist sono presenti i jar di Spring che a secondo del contesto vanno copiati nel classpath dell'applicazione. C'è da notare che sotto docs\reference\pdf è presente il file spring-reference.pdf che contiene un guida completa al framework.

CONOSCIAMO I CALCIATORI

A questo punto non ci resta che definire la classe rappresentante i singoli calciatori e lo storico dei voti ottenuti.

```
public class PlayerVO {
    public static final String RUOLO_PORTIERE="P";
    public static final String
                                RUOLO_DIFENSORE="D";
    public static final String
                           RUOLO_CENTROCAMPISTA="C";
    public static final String
                               RUOLO_ATTACCANTE="A";
    protected String nome=null;
    protected String ruolo=null;
    protected float mediaVoto=0;
    protected float mediaUltimoVoto=0;
    public float getMediaVoto() {
      return mediaVoto;
    }

    public void setMediaVoto(float average) {
     this.mediaVoto = average;
    }

    public float getMediaUltimoVoto() {
      return mediaUltimoVoto;
    }

    public void setMediaUltimoVoto(float
                                   lastAverage) {
     this.mediaUltimoVoto = lastAverage;
    }

    public String getNome() {
      return nome;
    }
```

LA CONFIGURAZIONE DEL DB

Questo progetto utilizza come database un DB Access memorizzato nel file fantacalcioKBS.mdb presente nel cd nella sottodirectory db del progetto. Per configurare il database Access è necessario:

- **Eseguire Data Sources (ODBC) presente in Settings/Control Panel/ Administrative Tools**
- **Scegliere User DSN/Add, selezionare Microsoft Access Driver(*.mdb) e poi il pulsante Finish**
- **In Data Source Name inserire la stringa KBS**
- **Nel riquadro Database, con il pulsante Select... selezionare il file .mdb in cui risiede il DB**
- **Confermare con OK nelle due form aperte**

```
public void setNome(String name) {
  this.nome = name;
 }

 public String getRuolo() {
  return ruolo;
 }
 public void setRuolo(String role) {
  this.ruolo = role;
 }
}
```

Come possiamo notare, si tratta di un semplice javabean che tramite i metodi set e get riesce a gestire, ruolo, media voto ed ultimo voto di ogni calciatore. Oltre a tali caratteristiche essa definisce anche delle costanti stringa per definire il set di ruoli possibili.
L'implementazione è abbastanza semplice, ma rappresenta un passo importante per 00l'implementazione del resto della nostra applicazione. Possiamo decidere qui molto su come implementare il tutto.

IL PATTERN INVERSION OF CONTROL

Si tratta di un modello di programmazione che recentemente sta incontrando un largo successo. La sua logica è semplice: un'applicazione è fatta di oggetti, ciascun oggetto può dipendere da altri oggetti, e ciascun oggetto può essere valorizzato in qualche modo. Secondo il pattern IoC esiste un "Container", un contenitore che porta in sé la conoscenza di come gli oggetti dipendono gli uni dagli altri e di come essi debbano essere valorizzati. È il Container a valorizzare gli oggetti e non loro a eseguire operazioni dirette gli uni con gli altri. Questo produce una logica di "disaccoppiamento" forte, ovvero consente di produrre applicazioni le cui classi non debbano necessariamente avere conoscenza di quelle da cui dipendono e questo provoca in molti casi vantaggi notevoli.

NOTA

ESEGUIRE L'APPLICAZIONE

Per far funzionare la nostra applicazione è necessario configurare il DB come descritto nell'apposito box laterale e successivamente eseguire il file run.bat nella sottodirectory bin del progetto. Il file XML letto da Spring è presente nella sottodirectory classes.

L'ENGINE, IL NOSTRO SELEZIONATORE

A questo punto, considerato che il KB ha solo il compito di recuperare i dati, concentriamoci sull'implementazione dell'Engine. Esso deve implementare l'interfaccia Engine e quindi tutti i suoi metodi pubblici.

```
public class Engine implements EngineI{

  protected int maxDifensoriTit=0;
  protected int maxCentrocampistiTit=0;
  protected int maxAttaccantiTit=0;
  protected final int maxDifensoriRis=3;
  protected final int maxCentrocampistiRis=2;
  protected final int maxAttaccantiRis=2;
  protected String schema=null;
  protected Comparator comparator=null;
  protected KBI kb=null;
```

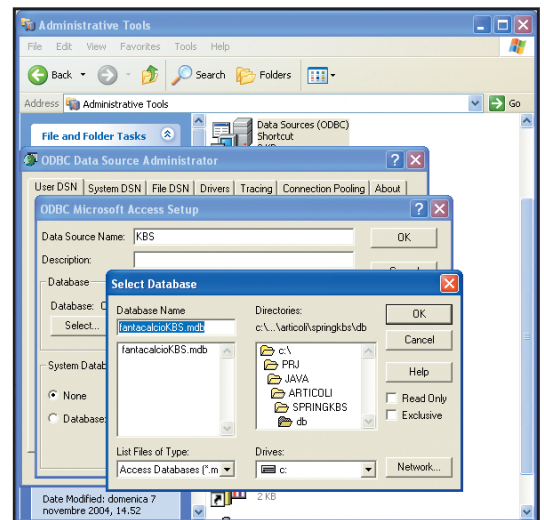



Fig. 2: ***I vari passi per la configurazione della connessione ODBC***

```

  public void setSchema(String schema) {
      this.schema=schema;
  }

  public void setComparator(Comparator co) {
     this.comparator=co;
  }

  public void setKb(KBI kb) {
     this.kb=kb;
  }

  protected ArrayList getPlayersByRole(ArrayList
                              players,String ruolo){
     ArrayList res=new ArrayList();
     PlayerVO p=null;
     for(int i=0;i<players.size();i++){
       p=(PlayerVO)players.get(i);
       if(p.getRuolo().equals(ruolo))
         res.add(p);
     }
     return res;
  }

   protected ArrayList getSubArrayList(ArrayList
                         ruoli,int start,int end){
      ArrayList x=new ArrayList(
                      ruoli.subList(start,end) );
      return x;
   }

   public ArrayList getTeam() {
     ArrayList team=new ArrayList(),
       allTeam=null,
       portieri=null,
      difensori=null,
      centrocampisti=null,
      attaccanti=null;
```

```
        allTeam=kb.getTeam();
maxDifensoriTit=Integer.parseInt
                              (schema.substrig(0,1));
maxCentrocampistiTit=Integer.parseInt
                              (schema.substring(1,2));
maxAttaccantiTit=Integer.parseInt
                              (schema.substring(2,3));
portieri=this.getPlayersByRole
                    (allTeam,PlayerVO.RUOLO_PORTIERE);
difensori=this.getPlayersByRole
                   (allTeam,PlayerVO.RUOLO_DIFENSORE);
centrocampisti=this.getPlayersByRole
              (allTeam,PlayerVO.RUOLO_CENTROCAMPISTA);
attaccanti= this.getPlayersByRole
                  (allTeam,PlayerVO.RUOLO_ATTACCANTE);
        Collections.sort(portieri,this.comparator);
        Collections.sort(difensori,this.comparator);
  Collections.sort(centrocampisti,this.comparator);
        Collections.sort(attaccanti,this.comparator);
          team.addAll
              ( this.getSubArrayList(portieri,0,1) );
          team.addAll
  (this.getSubArrayList(difensori,0,this.maxDifensor
                                                iTit));
        team.addAll(this.getSubArrayList
  ( centrocampisti, 0, this.maxCentrocampistiTit));
        team.addAll
                              (this.getSubArrayList
              (attaccanti,0,this.maxAttaccantiTit));
        team.addAll
               ( this.getSubArrayList(portieri,1,2));
        team.addAll
                              (this.getSubArrayList
                      (difensori,this.maxDifensoriTit,
        this.maxDifensoriTit+this.maxDifensoriRis));
        team.addAll
              (this.getSubArrayList(centrocampisti,
                           this.maxCentrocampistiTit,
                     this.maxCentrocampistiTit+this.
                                maxCentrocampistiRis));
        team.addAll (this.getSubArrayList
                     (attaccanti, this.maxAttaccant
                     iTit,    this.maxAttaccantiTit+
                     this.maxAttaccantiRis));
        return team;
    }
}
```

Come possiamo notare, oltre a definire variabili e costanti di supporto, è il metodo getTeam(...) il punto cruciale dell'intera applicazione. Al suo interno vengono prima fissati, in base allo schema scelto, quanti calciatori devono comporre i vari reparti.
Successivamente vengono ordinati, reparto per reparto, i calciatori in base al criterio (Comparator) scelto. Questa operazione viene effettuata tramite il metodo statico sort(...) di java.util.Collections; questo metodo accetta due parametri: il primo deve essere un oggetto che implementa java.util.List, mentre il secondo deve essere un Comparator (vedi apposito Riquadro). A questo punto basta mettere assieme i risultati raggiunti riservando i primi undici posti della nostra squadra ai titolari, a tal proposito vanno inseriti tanti calciatori per reparto quanti sono i posti ad essi riservati dallo schema scelto. Per quanto riguarda i sette calciatori destinati alla panchina abbiamo utilizzato una distribuzione fissa che prevede tre difensori, due centrocampisti e due attaccanti.

IL PRIMO MAIN

Utilizzando anche le classi di supporto analizzate nei riquadri, siamo pronti per scrivere il main della nostra applicazione per vederla finalmente funzionare.

```
public class JavaMain {
  public static void main(String [] ar){
    EngineI engine=new Engine();
    KBI kb=new MockKB();
    Comparator c=new MUVComparator();
    engine.setComparator(c);
    engine.setKb(kb);
    engine.setSchema("442");
    ArrayList fTeam=engine.getTeam();
    KBSUtil.stampa(fTeam);
  }
}
```

Come possiamo notare la sua implementazio-

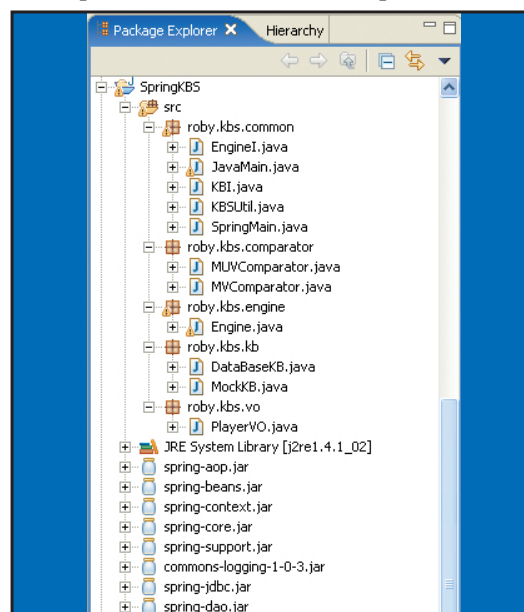



Fig. 3: *Le classi del progetto viste con Eclipse*

SETTERS O COSTRUTTORI?

Spring supporta due modi per effettuare la Dependency Injection (o Inversion of Control) tra gli oggetti di un'applicazione:

- **setter-based: è la modalità in cui i bean sono legati tra di loro utilzzando i metodi set<Proprieta>. Questa modalità è quella utilizzata in questo articolo e suggerita nella documentazione stessa.**
- **constructor-based: è la modalità in cui i bean sono legati tra di loro sfruttando il passaggio degli oggetti nei costruttori**

ne è davvero semplice; infatti dopo aver creato un Engine ed un MockKB, è necessario stabilire quale Comparator utilizzare per l'ordinamento dei calciatori e lo schema con cui schierare la nostra squadra. Dopo aver passato, per mezzo dei rispettivi metodi set, questi oggetti all'Engine è sufficiente invocare il metodo getTeam() per conoscere il risultato della decisione presa dall'Engine stesso. Eccoci giunti ad un punto molto importante di questo articolo, osservando la classe JavaMain ci accorgiamo che nonostatnte gli accorgimenti presi, per cambiare schema o Comparator o, come faremo da qui a poco, per utilizzare un KB un po' più sofisticato, è sempre necessario cambiare qualcosa nell'implementazione di questa classe. Tutto questo perché siamo obbligati a mettere insieme i cocci della nostra applicazione da codice Java tramite i metodi set esposti dalle classi utilizzate. Sarebbe comodo e utile scrivere solo le classi e legarle tra di loro con un file di configurazione. Questa caratteristica è la peculiarità principale di Spring.

E VENNE SPRING

È giunta l'ora di vederlo in azione. A tal proposito diciamo subito che uno dei maggiori punti di forza del suo impiego è la capacità di scrivere i vari oggetti di un'applicazione in modo del tutto disaccoppiato, ovvero quando si ha a che fare con Spring si scrivono i vari moduli e poi ci si affida al framework per metterli insieme. Questo modus operandi viene solitamente definito Inversion of Control o da taluni Dependency Injection. Spring applica l'Inversion of Control leggendo un file XML e deducendo da esso quali oggetti creare e come legarli tra di loro. Forti di queste poche considerazioni abbiamo capito che possiamo pensare a Spring come una Factory di classi dotata di una certa intelligenza. Per meglio comprendere il funzionamento di Spring analizziamo il main ed il file XML che utilizza Spring per comporre gli oggetti implementati nel nostro progetto.

```
public class SpringMain {

  public static void main(String [] ar){
    ClassPathXmlApplicationContext factory=
          new
        ClassPathXmlApplicationContext("kbs.xml");
    EngineI engine=(EngineI)
                  factory.getBean("engine");
    ArrayList fTeam=engine.getTeam();
      KBSUtil.stampa(fTeam);
    }
}
```

Rispetto al main visto in precedenza, possiamo notare che per prima cosa viene creato un oggetto istanza di ClassPathXmlApplicationContext nel cui costruttore passiamo il nome del file XML da utilizzare. Questo oggetto è un ApplicationContext, una delle tante factory utilizzabili con Spring, che carica i dati da un file compreso nel classpath. Inoltre non compaiono più i vari KBI e Comparator, infatti la loro creazione ed il loro passaggio nei metodi set di Engine vengono effettuati dal framework. Questo comportamento è guidato dal file kbs.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD
  BEAN//EN""http://www.springframework.org/dtd
                        /spring-beans.dtd">
<beans>
<bean id="kbImpl" class="roby.kbs.kb.MockKB"/>
<bean id="comparatorImpl"
      class="roby.kbs.comparator.MVComparator"/>
<bean id="engine"
                  class="roby.kbs.engine.Engine">
    <property name="kb" >
      <ref bean="kbImpl" />
   </property>
    <property name="comparator" >
      <ref bean="comparatorImpl" />
    </property>
    <property name="schema" >
      <value>433</value>
    </property>
</bean>
</beans>
```

0Il root tag di un file Spring è sempre <beans> ed al suo interno sono racchiusi uno o più tag <bean> che definiscono la composizione dei vari oggetti implementati nell'applicazione. In questo caso osserviamo che il primo tag è

```
<bean id="kbImpl" class="roby.kbs.kb.MockKB"/>
```

Il valore dell'attributo id definisce il nome con cui l'oggetto in questione può essere referenziato sia nel file XML, sia da codice Java. Il valore dell'attributo class definisce la classe dalla quale istanziare un oggetto; quindi, quando il framework legge questo tag, istanzia (utilizzando il costruttore senza parametri) un oggetto di tipo MockKB e lo lega ad un riferimento di tipo stringa che assume valore “kbImpl”. Il framework effettua un'operazione

simile quando legge il tag

```
<bean id="comparatorImpl"
      class="roby.kbs.comparator.MVComparator"/>
```

Infatti crea un oggetto MVComparator e lo lega all'id "comparatorImpl".
Un'operazione un po' più complessa viene effettuata dal framework quando legge il bean il cui id è "engine"; infatti oltre a creare un oggetto istanza di roby.kbs.engine.Engine deve effettuare un'operazione di setting per ogni tag property che contiene. Ogni property ha un attributo name, il suo valore identifica il metodo Java da utilizzare all'interno dell'oggetto creato.

```
<property name="kb" >
    <ref bean="kbImpl" />
</property>
```

Il precedente tag dice a Spring di invocare il metodo setKb e di passargli come argomento il bean creato con id valorizzato a "kbImpl".
Il secondo tag property indica a Spring di invocare il metodo setComparator di Engine e di passargli come parametro il bean il cui id è "comparatorImpl".
Nel terzo tag property vediamo che possiamo passare anche dei valori semplici (oggetti corrispondenti ai tipi predefiniti o String) nei metodi identificati dal tag stesso, infatti quando Spring legge questo tag invoca il metodo setSchema("442") sull'oggetto istanza di Engine.
Quando da codice Java recuperiamo l'oggetto istanza di Engine abbiamo l'oggetto già completo di tutti i suoi collaboratori (nel gergo di Spring si indica con collaborator un oggetto passato ad un altro tramite un metodo set), ecco perché non dobbiamo effettuare nessuna invocazione ai metodi setXXX da esso esposti.

I PRIMI VANTAGGI

Ovviamente Spring non è tutto qua. Ma già da questo breve esempio ne possiamo assaporare i vantaggi apportati dal suo utilizzo. Sicuramente possiamo affermare che utilizzare Spring spinge lo sviluppatore a scrivere codice migliore, cioè codice i cui moduli sono poco accoppiati. Questo fatto al giorno d'oggi è da tenere in forte considerazione, infatti questo approccio si presta molto al refactoring e allo sviluppo separato dei pezzi costituenti l'intera applicazione. La prototipazione è molto più veloce, nel nostro esempio quando scrivevamo il codice relativo all'Engine non avevamo fatto nessuna supposizione su come fosse implementata la base della cono-

Fig. 4: ***Il sito di riferimento del framework Spring***

scenza, lo schema ed il Comparator da utilizzare. Java ci consente di fare ciò attraverso le interfacce, ma Spring amplifica e rafforza questo approccio iniettando dall'esterno le dipendenze tra i vari oggetti. Questa caratteristica molte volte consente di cambiare il comportamento di un'applicazione senza cambiare il codice Java e senza ricompilarlo.
Infatti cosa sarebbe successo nella classe JavaMain se avessimo voluto adottare uno schema di gioco diverso? E se avessimo voluto utilizzare un altro Comparator per la scelta dei titolari?

MOCKKB UN'IMPLEMENTAZIONE BANALE

Molte volte, in fase si prototipazione e soprattutto di testing, vengono utilizzati degli oggetti, volutamente semplici ma sicuramente funzionanti, definiti Mock Object. Tali oggetti simulano il comportamento di oggetti molto più complessi mantenendo la stessa interfaccia verso chi li utilizza.

```
public    class MockKB implements
                              KBI{
   protected PlayerVO
             initPlayer(String nome,float
uv,float v,String ruolo){
      PlayerVO p=new PlayerVO();
      p.setNome(nome);
      p.setMediaUltimoVoto(uv);
      p.setMediaVoto(v);
      p.setRuolo(ruolo);
      return p;
   }
   public ArrayList getTeam() {
     ArrayList team=new ArrayList();
     team.add(this.initPlayer
                  ("Buffon", 7 ,6.3f,
          PlayerVO.RUOLO_PORTIERE));
  team.add(this.initPlayer("Peruzzi",7,6
          ,PlayerVO.RUOLO_PORTIERE) );
     team.add(this.initPlayer("Toldo",
 2.1f,6,PlayerVO.RUOLO_PORTIERE) );
     ...
     ... inizializzazione di altri oggetti
     ... PlayerVO
     ...
     return team;
   }
 }
```

In questo caso la creazione della rosa della squadra è definita tutta nel codice senza utilizzare file o DB per la loro persistenza. Questa implementazione poco elegante ed efficace non inficia il funzionamento del resto dell'applicazione.

nome	ruolo	mediavoto	ultimovoto
Baggio	A	7,5	8
Brocchi	C	5	5,5
Buffon	P	7,6	7
Cannavaro	D	7	7,5
Caracciolo	A	7	8
Chivu	D	5,5	6
Cordoba	D	6	6,5
Dainelli	D	6,5	7
Del Piero	A	7	8
Emerson	C	7	7,5
Ferrara	D	7	8
Fiore	C	7,5	8
Gilardino	A	6,5	7
Inzaghi	A	0,5	7
Maldini	D	7	8
Panucci	D	5	6
Peruzzi	P	6,5	7,89
Pirlo	C	6	7
Seedorf	C	7	8
Simic	D	6	6,5
Toldo	P	6	6,5
Toni	A	7	8,5
Vieira	C	7	7,5
Zanetti	C	6	7

Fig. 5: ***La tabella Access contenente la rosa della squadra***

SINGLETON E PROTOTYPE

Un altro grosso vantaggio nell'utilizzare Spring è la posssibilità di sfruttare la sua capacità di creare oggetti singleton. Quante volte in un'applicazione abbiamo l'esigenza di creare oggetti singleton? Di default ogni bean definito nel file XML è creato in singleton mode, quindi è un singleton condiviso da tutte le porzioni di codice che durante l'esecuzione del programma richiedono alla factory lo stesso oggetto tramite il suo id.

```
<bean id="unId"
        class="LaMiaClasse"
        singleton="false"/>
```

In questo esempio viene specificato che non si vuole fare uso di un singleton. Infatti, settando a false il valore dell'attributo singleton, si comunica alla factory di istanziare un nuovo oggetto ogni volta che si invoca il metodo getBean(...) della factory stessa; questa modalità viene definita prototype mode.

La risposta è sempre la stessa: modificare il codice della classe e ricompilarlo. Utilizzando SpringMain invece è sufficiente valorizzare opportunamente i vari tag XML! Oltre a cambiare il comportamento di un'applicazione, questo approccio torna utile anche quando si effettuano delle modifiche sostanziali ai singoli moduli. Nel nostro esempio abbiamo utilizzato un Mock Object per implementare la base della conoscenza, vediamo adesso quanto sia indolore cambiare la sua implementazione utilizzandone una che si affida al recupero dei dati da DB.

DATABASEKB

Per recuperare i dati dal DB utilizzeremo un database basato su MS Access, la classe DataBaseKB deve implementare KBI ed esporre il metodo getTeam().

```
public class DataBaseKB implements KBI{
  protected JdbcTemplate jt=null;

  public void setJdbcTemplate(JdbcTemplate jt){
    this.jt=jt;
  }

  public ArrayList getTeam() {
    List li=this.jt.queryForList
                          ("select * from votirosa");
    ArrayList aux=new ArrayList(li),
    res=new ArrayList();
    Map map=null;
    PlayerVO p=null;

    for(int i=0;i<aux.size();i++){
      map=(Map)aux.get(i);
      p=new PlayerVO();
      p.setNome((String)map.get("nome"));
      p.setRuolo((String)map.get("ruolo"));
      p.setMediaVoto
  (  ((Float)map.get("mediavoto")).floatValue()  );
      p.setMediaUltimoVoto
  (  ((Float)map.get("ultimovoto")).floatValue()  );
      res.add(p);
    }

  return res;
  }
}
```

La classe utilizza un oggetto istanza di JdbcTemplate definito nelle librerie JDBC di Spring. Questa classe semplifica molto la vita agli sviluppatori che sono soliti accedere ad un DB con le API JDBC di Java. In questo caso stiamo sfruttando un altro dei vantaggi rilevabili nell'utilizzo di Spring. Infatti questo framework implementa molti wrapper che racchiudono le operazioni di basso livello effettuabili con le librerie Java. In verità Spring implementa molte classi, anche più sofisticate di JdbcTemplate, per l'interazione con i DB; abbiamo voluto utilizzare questa classe per non complicare troppo il primo incontro con questo splendido framework.
Come possiamo notare l'oggetto accetta la query sql come stringa nel metodo queryForList(...) e restituisce un oggetto List dove vengono memorizzati i risultati. Ogni elemento della lista è un oggetto che implementa java.util.Map e che emula il comportamente di un java.sql.ResultSet.
A livello implementativo bisogna dire che JdbcTemplate accetta un parametro inerente

il datasource da utilizzare per la connessione. Questo parametro però lo si può iniettare dal file XML; per utilizzare il DataBaseKB bisogna integrare il file XML visto in precedenza con i tag XML mostrati di seguito.

```
<bean id="ds"
      class="org.springframework.jdbc.datasource.
                      DriverManagerDataSource">
<property name="driverClassName" >

<value>sun.jdbc.odbc.JdbcOdbcDriver</value>
</property>
<property name="url" >
    <value>jdbc:odbc:KBS</value>
</property>
</bean>
<bean id="jt"
          class="org.springframework.jdbc.core.
                              JdbcTemplate">
<property name="dataSource" >
     <ref bean="ds" />
</property>
</bean>
<bean id="kbImpl"
              class="roby.kbs.kb.DataBaseKB">
<property name="jdbcTemplate" >
    <ref bean="jt" />
</property>
</bean>
```

Questi tag vanno sostituiti al tag che definiva l'utilizzo del MockKB. In questo caso viene creato un oggetto DataSource i cui parametri di inizializzazione vengono fissati nel file XML stesso. In particolare il primo parametro viene valorizzato con la stringa rappresentante il driver JDBC-ODBC. Il secondo parametro indica la url del database ODBC. Il secondo bean definisce la creazione di un oggetto JdbcTemplate, mentre la property con name "datasource" equivale all'invocazione del metodo setDataSource al quale viene passato l'oggetto DataSource creato secondo quanto detto del tag precedente. Infine l'ultimo tag implica la creazione da parte del framework di un oggetto DataBaseKB al quale successivamente viene passato, tramite il metodo setJdbcTemplate(...), l'oggetto definito nel tag con id valorizzato a "jt".

CONCLUSIONI

Abbiamo imparato ad utilizzare il framework Spring valutandone i notevoli vantaggi derivanti dal suo utilizzo.
L'applicazione implementata non vuole essere una soluzione completa, ma sicuramente può essere utilizzata come punto di partenza per l'implementazione di un sistema esperto per la gestione di una squadra del Fantacalcio. C'è da ripetere per l'ennesima volta che in questo articolo abbiamo analizzato solo i principi funzionali di Spring, framework che si distingue anche per molte altre caratteristiche. La cosa che bisogna capire da subito è che il framework sfrutta l'inversion of control in tutti i contesti in cui può essere applicato. Alcuni di questi contesti sono le applicazioni web che seguono il pattern MVC, l'AOP (Aspect Oriented Programming), l'integrazione di alcuni framework per la persistenza come Hibernate, JDO e iBatis, l'integrazione di tecnologie per il remoting come RMI ed EJB Session e molto altro ancora.
L'alto grado di disaccoppiamento offerto da IOC porta in se il comodo vantaggio di dovere scrivere dei componenti che presi singolarmente compiono una solo singola operazione. Nessun componente dipende direttamente da un altro, ma tutti fanno capo a un contenitore centralizzato che garantisce la comunicazione fra i vari elementi dell'applicazione. L'altro grande vantaggio di Spring risiede nel fatto che non si tratta di un framework invasivo, ovvero non implementa nuovi ulteriori framework, invece sfrutta quelli già esistenti in modo da renderli omogenei e facilmente interscambiabili. In questo modo il programmatore non ha moltissime nuove nozioni da imparare per usare Spring, e la curva di apprendimento risulta piuttosto bassa.

Roberto Sidoti

COMPARATOR, IL CRITERIO DI ORDINAMENTO

In questa applicazione vengono utilizzati due tipi di comparator customizzati per ordinare in modo decrescente i calciatori in base alla loro media voto o in base alla media relativa all'ultmo voto. Giusto per capirne l'implementazione ne riportiamo di seguito solo uno, l'implementazione dell'altro è molto simile.

```
public class MUVComparator
              implements Comparator{
    public int compare(Object player1,
                        Object player2){
      PlayerVO p1=(PlayerVO)player1;
      PlayerVO p2=(PlayerVO)player2;
      Float pf1=new
        Float(p1.getMediaUltimoVoto());
      Float pf2=new
        Float(p2.getMediaUltimoVoto());
      return (-1) * pf1.compareTo(pf2);
    }
}
```

Come ogni comparator deve implementare un metodo compare che accetta due Object. L'implementazione è molto semplice e necessita di poche spiegazioni, in particolare il metodo restituisce un valore intero appartenente all'insieme {-1,0,1}. Come possiamo notare i tre valori vengono moltiplicati per –1 in quanto l'ordine che si vuole utilizzare è quello decrescente, quindi i tre valori in questo caso determinano: p1>p2, p1=p2, p1>p2.

L'AUTORE
Roberto Sidoti è ingegnere informatico. Attaulmente si occupa di servizi VoIP basati sul protocollo SIP presso Telecom Italia LAB come consulente esterno. I suoi maggiori interessi riguardano le applicazioni distribuite sviluppate con tecnologia J2EE. Da marzo 2005 si occupa dello sviluppo e della gestione del progetto GeiNuke (www.hostingjava.it/-geinuke/).

È IN ARRIVO XAML NUOVO E SEMPLICE

È ARRIVATO IL MOMENTO DI DIRE ADDIO A MIGLIAIA E MIGLIAIA DI RIGHE DI CODICE. CON XAML SVILUPPEREMO INTERFACCE PER APPLICAZIONI STANDALONE CON LA STESSA SEMPLICITÀ CON CUI SVILUPPIAMO PER IL WEB. SEMPLICE E POTENTE

XAML è l'acronimo di **eXtensible Application Markup Language** ed è, come il nome fa intuire, un linguaggio basato su XML per la creazione di applicazioni. XAML è, per essere precisi, uno dei tanti modi in cui è possibile creare interfacce per Windows, sfruttando un toolkit noto come **WinFX**, che racchiude al proprio interno tutte le tecnologie necessarie a creare presentation layer, piuttosto che funzionalità di comunicazione o workflow.
A differenza di quello che uno sviluppatore poco attento al mercato potrebbe aspettarsi, WinFX sarà disponibile non solo per Windows Vista, la prossima edizione di Windows Client, ma anche per **Windows XP** e **Server 2003**, rendendo di fatto le applicazioni disponibili per tutta la piattaforma Windows. Attualmente esiste una CTP, una preview, ma a giorni è attesa la beta 2. Il rilascio definitivo è per la fine del 2006, contestualmente al lancio sul mercato di Windows Vista.

C'È SEMPRE IL .NET FRAMEWORK DIETRO A WINFX

Come è logico aspettarsi, alla base di WinFX (e quindi di XAML) c'è il .NET Framework. WinFX, infatti, è un'evoluzione dell'attuale versione 2.0, con in più alcune significative funzionalità.
Ovviamente rispetto alla 2.0, sul mercato da alcuni mesi, ci saranno degli assembly specifici, che andranno a contenere tutte quelli classi pensate appositamente per le tre componenti chiave di WinFX:

- **Windows Presentation Foundation**: anche noto con il nome in codice di Avalon, è la parte che in WinFX contiene tutte le funzionalità destinate al presentation layer;
- **Windows Communication Foundation**: rappresenta tutte le funzionalità dedicate alla comunicazione, con un sotto-framework completamente unificato in grado di garantire lo stesso modello ad oggetti a prescindere dal transportation layer utilizzato, sia MSMQ piuttosto che web services. In beta è conosciuto con il nome in codice di Indigo;
- **Windows Workflow Foundation**: fornisce l'infrastruttura necessaria alla creazione di applicazioni che implementino logica basata su Workflow.

Allo stato attuale WPF è quello che attira maggiormente gli sviluppatori e sul quale ci soffermeremo, dato che XAML è semplicemente un linguaggio attraverso il quale si possono creare applicazioni che poi gireranno sotto WPF, con l'ausilio del .NET Framework.

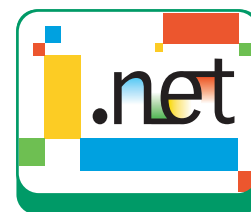
PERCHÉ UNA RIVOLUZIONE NELLA PIATTAFORMA WIN32?

La risposta più scontata è che per oltre venti anni la piattaforma è rimasta sempre la stessa, basata su GDI ed un po' di librerie User. Quindi i suoi limiti sono essenzialmente dettati dell'essere stata introdotta in condizioni diverse da quelle che attualmente regolano il mercato. La maggior parte delle applicazioni oggi in circolazione è infatti basata su GDI e non ci sono limitazioni particolari. Come chiunque abbia creato applicazioni per Windows sicuramente sa, a parte il fatto di essere fortemente orientato al pixel nella fase di rendering. La motivazione principale di un rinnovamento così marcato è che il livello dell'hardware odierno consente di fare cose

che, quando GDI è stato pensato, non erano nemmeno immaginabili.
Oramai RAM, processori e dischi fissi, per non parlare delle schede video, garantiscono performance di tutto rispetto. La verità è che anche se abbiamo un hardware molto spinto, la maggior parte delle applicazioni attuali continuano ad usarlo (quasi) come quello di dieci anni fa.
Per capirci, se utilizzate una scheda video con 256 MB di RAM con Office, rispetto ad una dotata di un solo MB, non noterete nessun vantaggio dall'aver speso qualche euro in più. Molto probabilmente l'unico sistema attualmente disponibile per sfruttare al massimo il proprio hardware, ironia della sorte, sono i giochi.
Tuttavia ci sono scenari in cui avere una potenza maggiore, che tra l'altro è anche già disponibile, potrebbe aiutare la user experience, ovvero la comodità con la quale un utente utilizza una data applicazione.
E poi, semplicemente, ormai video, fotografie, musica e più in generale contenuti multimediali sono alla base di moltissime attività, dunque poter offrire una serie di servizi orientati al multimedia è un vantaggio da non trascurare. Le interfacce a riga di comando, tanto per capirci, hanno smesso di avere senso.
Con le nuove tecnologie, come quelle offerte dagli **Home Teather PC** (HTPC), di cui Windows XP Media Center Edition è un ottimo esempio, gli utenti si sono sempre più abituati a vedere il computer non solo come uno strumento di lavoro, ma anche di piacere.
E la prima cosa che si nota quando si passa dalla TV al computer è che quest'ultimo, spesso, ha un'usabilità minore derivante dal fatto che le interfacce sono più complesse e meno accattivanti dal punto di vista estetico. Mentre in realtà la potenza e la versatilità di un computer dovrebbe garantire l'esatto contrario.
In realtà per chi sviluppa utilizzando il .NET Framework esiste un'estensione di GDI, nota come **GDI+**, che consente di sfruttare un engine grafico dove sono disponibili alcune primitive, con una resa grafica migliore, che in accoppiata con **DirectX** e **Direct3D** rende più semplice sfruttare al meglio l'hardware della scheda grafica.
Tuttavia ciò che manca ancora è **una piattaforma comune**, che abbia linee guida ben definite, consenta di creare applicazioni dal look curato, semplicemente e sfruttando al massimo l'hardware a disposizione e possibilmente senza vincolarsi ad un tipo di device particolare.
Windows Presentation Foundation nasce proprio per questo.

COSA FA E COSA NON FA WPF

WPF è in grado di rendere possibile la creazione di qualsiasi tipo di applicazione:
- applicazioni "classiche";
- applicazioni vettoriali 2D, anche animate;
- applicazioni vettoriali 3D, anche animate;
- applicazioni con video, audio, immagini;
- applicazioni documentali;
- applicazioni con navigazione, stile web.

WPF consente di definire oggetti vettoriali, disegnati all'interno di un'area di lavoro, esattamente come fareste con un qualsiasi tool di grafica. Con il vantaggio che ciò che create non rimane solo un'immagine, ma può essere programmato e reso dinamico: può interagire con un sito remoto, mostrare informazioni recuperate dal disco locale, aggregare informazioni.
Ed il vantaggio di utilizzare la **grafica vettoriale** anziché **quella raster** è che consente di rendere più facile la fruizione dei contenuti, a prescindere dalla risoluzione del dispositivo.
In più questo vuol dire che, praticamente, qualsiasi programma che supporti la grafica vettoriale, potenzialmente, potrebbe offrire un'esportazione in XAML, che è il formato attraverso il quale vengono definiti gli oggetti all'interno di WPF, dato che si tratta solamente di definire dentro un file di testo una serie di direttive che poi, una volta eseguite con WPF, diventeranno oggetti vettoriali sullo schermo dell'utente.
Per la prima volta nella sua storia, Microsoft ha annunciato una suite di tool espressamente pensata per i designer, all'interno della famiglia di prodotti denominati **Expression**.
http://www.microsoft.com/products/expression/en/default.aspx
Ne esiste uno specifico per XAML, nome in codice **Sparkle**, che ha l'obiettivo di rendere possibile il lavoro di un designer all'interno di un team di sviluppatori, scenario molto difficile sfruttando gli attuali tool. Il designer infatti trova un ambiente che gli consente di concentrarsi espressamente sul lato grafico dell'applicazione, generando però un output in formato XAML, che poi gli sviluppatori potranno animare, come di consueto.
Finora, infatti, siamo stati abituati ad avere interfacce stile **gestionale degli Anni 90**, con prevalenza di grigio o, peggio ancora, abbondanza di colori accostati casualmente. Con

l'avvento di un modello del genere sarà finalmente possibile avere applicazioni funzionali ed anche dal giusto appeal grafico.

UN PRIMO SGUARDO A XAML

Cos'è esattamente XAML, oltre che un linguaggio basato su XML?
È essenzialmente un approccio diverso allo sviluppo di interfacce, attraverso il quale anziché definirle attraverso il designer di un tool, lo si può fare anche sfruttando dei marcatori testuali.
Per chi ha dimestichezza con ASP.NET, o più in generale con HTML, questo approccio risulterà del tutto naturale. Lo dimostra l'esperienza: poter definire un layout utilizzando un linguaggio di markup facilita la creazione delle applicazioni, sia da parte di umani che da parte di tool grafici.
L'esperienza ed il grande successo di ASP.NET, poi, hanno consentito di poter investire in questa direzione senza fare un salto nel buio, ma con il suffragio dato da anni di sviluppo.
D'altra parte XML, di cui XAML è solamente un "dialetto", è una tecnologia pervasiva, ma tutto sommato facile: è semplicemente un **file di testo ASCII**.
Questo rende XAML praticamente interoperabile con qualsiasi tool di sviluppo, tanto è vero che, quando WinFX sarà rilasciato, uno dei tool principali con il quale sarà possibile creare le interfacce sarà disponibile proprio grazie ad **Adobe**. E già adesso esistono vari tool, anche di terze parti, che convertono dai più diffusi formati vettoriali in XAML. Ad esempio, su *http://lab.aspitalia.com/15/EmfToXamlBeta.aspx* potete trovarne uno che vi consente di convertire clipart dal formato EMF, usato anche da Office, in XAML. Provatelo perché così capirete dove sia la potenza di un approccio del genere.
XAML è anche e soprattutto un insieme di classi managed, ovvero eseguite all'interno del .NET Framework, nel caso specifico WinFX. Queste classi si trovano all'interno del namespace **System.Windows.Forms** e sono contenuti all'interno di diversi assembly, tra cui 4 assembly specifici:

- PresentationFramework.dll
- PresentationCore.dll
- PresentationUI.dll
- WindowsBase.dll

Se avete mai creato una finestra contenente un pulsante con C++, ma anche VB 6 o VB.NET, ecco come fare la stessa cosa usando XAML:

```
<Window xmlns="http://schemas.microsoft.com/
        winfx/avalon/2005"><Button Width="100"
                Height="30">Cliccami</Button>
</Window>
```

Semplicemente eseguendo questo file all'interno di un computer che abbia WinFX installato, il risultato, come testimonia l'immagine qui a fianco, è una finestra con un pulsante.

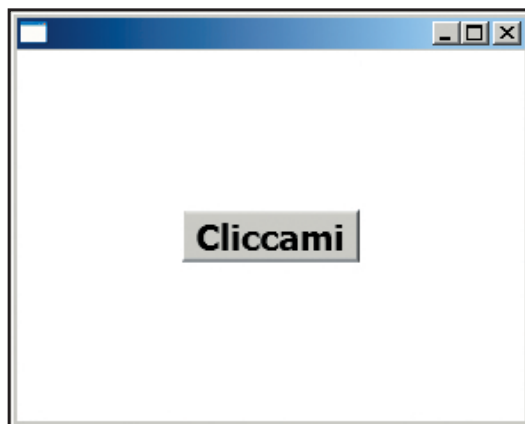



Fig. 1: un semplice pulsante generato con XAML

Come si può notare, tra l'altro, i nomi dei singoli oggetti ricordano molto da vicino ciò che poi effettivamente saranno le funzionalità.
In effetti gli oggetti sono inseriti in maniera dichiarativa all'interno del file XAML e poi convertiti dal motore, in fase di esecuzione, in istanze del corrispondente oggetto. E così nel nostro esempio non faremo altro che creare un'istanza della classe Window, che rappresenta la finestra, e poi una di quella Button, che rappresenta il pulsante, aggiungendo quest'ultimo alla finestra.
Rispetto ad un modello programmatico, quello dichiarativo ha il vantaggio di essere molto più immediato.
Dunque, per fissare meglio i concetti, un tag rappresenta una classe, mentre ogni attributo una proprietà di essa.
Nella costruzione delle GUI XAML sostituisce al 100% il codice, come ad esempio avviene all'interno di una tipica WinForm creata con Visual Studio. Ed il bello è che tutto ciò che è possibile fare da XAML, è sicuramente possibile farlo anche da codice, certo con uno sforzo maggiore.

WEB BROWSER APPLICATION

WPF non si ferma solo alle normali applicazioni Windows che oggi conosciamo. Abbiamo infatti la

possibilità di offrire all'utente l'esperienza web, alla quale ormai tutti siamo abituati. Sostituendo Window con NavigationWindow la finestra sarà dotata dei classici comandi avanti ed indietro che tutti i browser offrono.
Ogni pagina nella quale andremo a navigare sarà rappresentata da un file XAML, avente come tag root <Page />.
In questo modo uniremo le migliori caratteristiche del web con le applicazioni rich-client, sfruttando al massimo le potenzialità della macchina.
Come una qualsiasi pagina web, infatti anche questa tipologia di applicazioni può essere distribuita tramite un server web, mediante un semplice click. Un file wba (web browser application) andrà a contenere le informazioni necessarie a stabilire chi è che distribuisce l'applicazione, oltre alla lista dei files necessirari al corretto funzionamento della stessa.
Alla pressione del link, entra in gioco **ClickOnce**: all'interno del .NET Framework 2.0 esiste un componente pensato per il deployment delle applicazioni che prende in carico questo file, recupera il manifest, che indica i permessi minimi che l'applicazione necessita, ed installa nella cache di Internet Explorer l'assembly, che è un eseguibile (.exe).
Ogni qualvolta poi l'utente riavvia l'applicazione, verrà effettuato un controllo sulla presenza o meno di una nuova versione e, nel caso fosse presente, viene effettuato il relativo aggiornamento.
Questo ci permette di creare applicazioni online, poiché Internet Explorer ospita l'applicazione tramite un processo esterno, ma il risultato visivo è esattamente lo stesso di una normale pagina web, come potete vedere nell'immagine allegata.

Fig. 2: ***Un'applicazione "Ospitata" all'interno di Internet Explorer***

GLI ELEMENTI DI UN FILE XAML: SHAPES, CONTROLS E PANELS

Analizzate brevemente le varie tipologie di applicazioni, passiamo alle famiglie di elementi che abbiamo a disposizione all'interno di WPF.
Primi tra tutti ci sono gli **Shapes**, oggetti vettoriali primitivi indispensabili per disegnare gli elementi principali, quali linee, rettangoli, poligoni ed ellissi. Per creare ad esempio un rettangolo di colore rosso e bordo blu dobbiamo usare la classe Rectangle. Tradotto in codice XAML diventa:

```
<Rectangle Width="200" Height="150"
        Fill="Red" Stroke="Blue" />
```

Non sempre però i valori delle proprietà sono tipi semplici, come interi o stringhe, perciò è disponibile una seconda sintassi, nella forma nomeclasse.proprietà, da scrivere come tag invece che come attributo. Ecco un esempio rende subito l'idea:

```
<Rectangle Width="200" Height=
        "150" Stroke="Blue">
    <Rectangle.Fill>
        <LinearGradientBrush>
    <LinearGradientBrush.GradientStops>
<GradientStop Color="Yellow" Offset="0" />
<GradientStop Color="White" Offset="1" />
    </LinearGradientBrush.GradientStops>
        </LinearGradientBrush>
    </Rectangle.Fill>
</Rectangle>
```

Questo codice definisce come colore di riempimento un oggetto complesso di tipo **LinearGradientBrush**, che rappresenta un gradiente che parte dal giallo e finisce al bianco.
Come si può notare in questo caso le proprietà da specificare sono diverse e quindi questa sintassi è fatta apposta proprio per scenari del genere.
A questi elementi possiamo anche affiancare immagini, video, suoni e oggetti 3D.
La seconda famiglia è quella dei **Controls**: elementi che interagiscono con l'utente, come Button, TextBox, Menu, ListBox e moltissimi altri.
Si differenziano dagli altri per il fatto che dispongono di uno o più Content, per la capacità di ricevere input o di essere selettori di elementi.
Una differenza importante, infatti, quando si sviluppa per WPF è nel **meccanismo di input**. Gli eventi che sono scatenati sono indipendenti dal fatto che "l'inserimento" provenga da tastiera, da mouse o da una penna di un TabletPC, cambiando quindi l'approccio al consumo degli eventi ed aprendo la strada a future nuove forme di input.
Tutta l'architettura Win32 è fortemente basata sui messaggi, mentre in una finestra WPF questo concetto non esiste, sostituito da un sistema di **routing dell'evento**.
Al contrario di ciò che avviene attualmente con le WinForms, un evento non raggiunge subito il con-

trollo destinatario, ma parte dall'elemento padre principale, scorrendo tutto l'albero dei figli fino ad arrivare al controllo finale.
Questo processo di "**tunneling**" permette di dotare i controlli di pre-eventi, come ad esempio PreviewMouseDown, per intervenire fermando l'evento qualora ci sia la necessità durante il suo percorso di notifica.
Una volta giunto a destinazione, l'evento subisce il percorso contrario, di "**bubbling**", ritornando all'elemento padre, e scatenando questa volta i normali eventi, come MouseDown.
Anche se a prima vista può sembrare più complesso come approccio per chi è abituato ad utilizzare i messaggi, questo nuovo modello consente di gestire l'input e gli eventi ad esso associati in maniera molto più potente.
Ultimi tra gli elementi visivi, ma non per questo meno importanti, sono i **Panels**. Contenitori degli elementi appena descritti, agiscono sulla dimensione e posizione dei loro figli.
Ecco un esempio:

```
<Canvas>
  <Button Canvas.Left="10"
                    Canvas.Top="10">Cliccami</Button>
</Canvas>
```

Tra questi il più intuitivo da usare è **Canvas**, che posiziona gli elementi in base ad una precisa coordinata X e Y.
Ce ne sono poi altri che agiscono secondo modalità differenti.
DockPanel, ad esempio, ancora gli elementi alla finestra, mentre **StackPanel** allinea tutti i figli orizzontalmente o verticalmente su una stessa linea. **Grid**, invece, permette di allineare in una ipotetica griglia i controlli figli.
L'esempio seguente definisce quattro pulsanti, ognuno posizionato in una cella differente di una tabella composta da due righe e due colonne:

```
<Grid ShowGridLines="True">
            <RowDefinition Height="100" />
            <RowDefinition Height="100" />
            <ColumnDefinition Width="150" />
            <ColumnDefinition Width="150" />
            <Button Grid.Column="0" Grid.Row="0"
                              Content="Button1" />
            <Button Grid.Column="1" Grid.Row="0"
                              Content="Button2" />
            <Button Grid.Column="0" Grid.Row="1"
                              Content="Button3" />
            <Button Grid.Column="1" Grid.Row="1"
                              Content="Button4" />
</Grid>
```

Il risultato è visibile nell'immagine e, come si può notare, a differenza delle tabelle HTML la definizione della struttura della tabella è separata rispetto agli elementi che essa contiene, per rendere più semplice un eventuale cambiamento del contenuto.

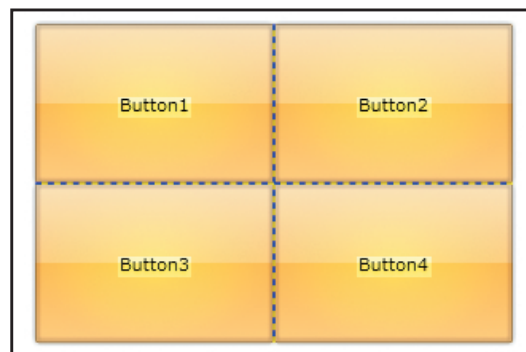



Fig. 3: la tabella è stata ottenuta tramite sintassi XAML

Da sottolineare che in base al contenitore nella quale l'elemento si trova, le informazioni da specificare cambiano.
Nelle attuali WinForms, così come è possibile anche in HTML, seppur non consigliato, ma soprattutto come siamo abituati a fare nell'uso comune, si è soliti posizionare un elemento in base a coordinate X e Y.
Questo approccio è però corretto se l'elemento si trova in un Canvas, specificando le proprietà Canvas.Left e Canvas.Top, ma in un Grid le informazioni che servono devono specificare in quale colonna e riga deve posizionarsi l'elemento.
Nell'esempio precedente lo facciamo proprio tramite le proprietà Grid.Column e Grid.Row. Ancora, per un DockPanel la proprietà DockPanel.Dock indica l'ancoraggio dell'elemento rispetto agli altri, dato che questi ultimi hanno la caratteristica di "legarsi" su uno dei lati al proprio contenitore.
Questo sistema è infatti disponibile tramite un meccanismo di dipendenze: ogni elemento eredita da una classe **DependencyObject**, che gestisce uno o più DependencyProperty. Queste proprietà fanno sì che sebbene venga specificato l'attributo, nell'esempio Grid.Column, sul Button, l'informazione venga comunque salvata sul Grid stesso.
Insomma, l'inserimento di Grid.Column viene tradotto dal parser in una chiamata al metodo statico Grid.SetColumn(elemento, valore), e come si può notare è ciò fortemente legato alla griglia che poi in realtà contiene il controllo

DIAMO UN TOCCO DI STYLING

I controlli sono "brutti" se non gli applichiamo un certo stile, che possa renderli più gradevoli dal punto di vista estetico.

NOTA

PER CONTINUARE

Di sicuro, come avrete notato con questa breve introduzione, XAML promette di rendere lo sviluppo di applicazioni per Windows ancora più potente, grazie alle nuove funzionalità offerte, ma soprattutto più gradevoli da vedere: anche l'occhio vuole la sua parte, diceva un proverbio, e questo è quanto più vero quando parliamo di uno schermo!

Il concetto è simile a ciò che i CSS (Cascaded Style Sheet) rappresentano per il web, ma le potenzialità sono maggiori perché c'è una scelta più ampia di opzioni.
In WPF è possibile definire due tipi di risorse: statiche o dinamiche.
Entrambe sono esposte dalla proprietà Resources, che ha ogni elemento, e si differenziano dal fatto che possono mutare o meno in fase di runtime.
Nelle risorse possiamo definire una serie di oggetti che poi possono essere utilizzati dall'applicazione e proprio tra gli oggetti che possiamo inserire ce n'è uno rappresentato dalla classe Style.
Se a questo oggetto aggiungiamo un valore per l'attributo Key, possiamo indicare uno o più controlli che andranno ad usare questo particolare stile. Specificando invece un valore per TargetType, diciamo al motore di applicare lo stile ad una particolare tipologia di elemento.
Aiutiamoci con un esempio, utilizzando il solito bottone. Procediamo con il definire tra le risorse della pagina uno stile, che poi utilizzeremo nel button:

```
<Page.Resources>
        <Style x:Key="style1">
        <SetterProperty="Button.Background"
                                Value="Blue" />
        <Setter Property="Button.Foreground"
                                Value="White" />
        </Style>
</Page.Resources>
<Button Style="
        {StaticResource style1}">Default</Button>
```

Style possiede una collezione di setters per ridefinire le proprietà del Button e non si limita solo a cambiarne l'aspetto in modo statico, ma anche al variarsi di una proprietà.
Tramite un meccanismo noto come trigger, possiamo agganciarci ad una proprietà e utilizzare di nuovo i setter per cambiarne l'aspetto grafico.
Per esempio se al passaggio del mouse si vuole modificarne i colori dell'oggetto, è sufficiente agganciarsi alla proprietà booleana IsMouseOver, così:

```
<Style x:Key="style1">
        <Setter Property="Background"
                                Value="Black" />
        <Setter Property="Foreground"
                                Value="White" />
        <Style.Triggers>
                <Trigger
        Property="Button.IsMouseOver" Value="true">
        <SetterProperty="Button.Background"
                                Value="Red" />
                </Trigger>
        </Style.Triggers>
</Style>
```

Il risultato è visibile nell'immagine a fianco, insieme ad un esempio complesso di restyling del pulsante.

Fig. 4: ***Modificare il layout attraverso un evento è semplicissimo***

Ciò che rende lo styling di WPF molto potente è la capacità di ridefinire qualsiasi proprietà che l'elemento possiede, senza limiti, e con un modello per intercettare gli eventi, basato su trigger, davvero molto semplice da imparare.
A tutto questo si aggiunge la facoltà di sfruttare le risorse dinamiche, per variare tutti gli style a runtime, e fornire per esempio funzionalità di skin alle nostre applicazioni.
Il pulsante mostrato nel primissimo esempio sembrerebbe a tutti gli effetti un normale pulsante in stile Windows Classic. Se l'applicazione fosse stata avviata su XP il look sarebbe stato invece il classico tema Luna.
In realtà è solo apparenza, perché si tratta di un'emulazione del look grafico fornito dalle Win32 che le attuali applicazioni Windows usano.
Poiché in WPF queste API non vengono usate, è stato replicato interamente, ma come styling di XAML, ed inserito in assemblies separati contenenti solo codice XAML di questo tipo.
Gli assemblies in questione sono PresentationFramework.Classic.dll e PresentationFramework.Luna.dll, distribuiti insieme a WinFX.

TRASFORMAZIONI E ANIMAZIONI

WPF è anche molto di più, perché la nuova architettura descritta in questo articolo è volta a sfruttare al massimo anche la capacità della scheda video.
Una moderna scheda video, infatti, è ormai munita di un processore in grado di compiere trasformazioni geometriche 2D e 3D o appli-

NOTA

Se l'articolo vi ha incuriosito e volete applicare ciò che vi è stato presentato, potete partire dallo scaricare da MSDN la CTP di WinFX e l'SDK attualmente disponibili per gli abbonati.
Dal sito http://msdn.microsoft.com/winfx/ si trovano inoltre le estensioni per Visual Studio 2005 per aggiungere i templates di progetto WPF e il supporto all'intellisense di XAML.
La versione specifica di Visual Studio pensata per XAML, infatti, non uscirà prima dell'anno prossimo, con buona probabilità, e nel periodo intermedio ci si dovrà "accontentare" di VS 2005, attraverso estensioni.

care effetti e dispone di memoria dedicata per il buffering delle immagini e la relativa elaborazione. Ecco quindi che ogni elemento di WPF è fornito delle proprietà **RenderTransform** e **LayoutTransform**. Il loro scopo è applicare delle trasformazioni all'elemento, che permettano di ruotarlo, ridimensionarlo, traslarlo o più in generale applicargli una matrice.
Il codice in XAML per ruotare ad esempio di 45 gradi un'immagine è semplice e immediato:

```
<Image Source="aspitalia.gif" Opacity="0.5"/>
<Image Source="aspitalia.gif">
        <Image.RenderTransform>
                <RotateTransform Angle="45"
                                Center="182, 35" />
        </Image.RenderTransform>
</Image>
```

Il risultato è visibile nella figura a fianco.
A questa funzionalità si affianca la possibilità

Fig. 5: **Ecco come appare il testo ruotato di 45**

di animare nel tempo qualsiasi elemento o proprietà presente.
Una serie di oggetti Animation sono in grado di variare ogni tipo managed in funzione della durata, delle accelerazioni o decelerazioni, su un preciso evento o all'infinito. Si tratta di classi che mutano i tipi Double, Integer, Byte, String, Color, Matrix, Point, Vector e così via.
Le animazioni sono legate ad uno **Storyboard**, un oggetto che le coordina, e che si avvia o si ferma in funzione della TriggerAction alla quale appartiene.
Negli Style abbiamo visto come la proprietà Triggers contegga un oggetto Trigger per dipendere da una proprietà, ma come sia anche disponibile un EventTrigger, che si aggancia ad un evento e può contenere i TriggerAction che avviano, sospendono, fermano una Storyboard.
Se vogliamo, per esempio, variare la dimensione di un pulsante quando ci si posiziona sopra con il mouse, dobbiamo intercettare l'evento MouseEnter, vviare un'animazione che ne modifichi la proprietà Width del button, in questo modo:

```
<Button Content="Pulsante!" Width="120">
        <Button.Triggers>
                <EventTrigger
                RoutedEvent="Button.MouseEnter">
                        <EventTrigger.Actions>
                        <BeginStoryboard>
                        <Storyboard>
<DoubleAnimation
        Storyboard.TargetProperty="(Button.Width)"
        AutoReverse="true" RepeatBehavior="Forever"
                To="160" Duration="0:0:1" />
</Storyboard>
                        </BeginStoryboard>
                </EventTrigger.Actions>
                </EventTrigger>
        </Button.Triggers>
</Button>
```

Nell'immagine si vede il pulsante nello stato iniziale ed in quello successivo, dopo un secondo dall'avvio dell'animazione.
L'oggetto DoubleAnimation, infatti, varia la proprietà Width del pulsante, indicata tramite

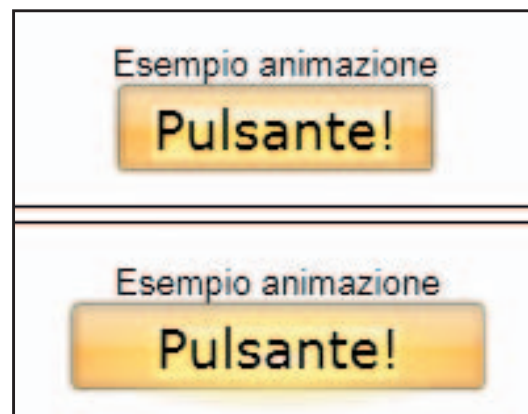



Fig. 6: **Un esempio di animazione, nell'immagine il pulsante nello stato originale e come appare dopo 45 secondi**

l'attributo TargetProperty. Tramite l'attributo To indichiamo il punto finale dell'animazione, mentre con Duration la sua durata e con AutoReverse invertiamo la marcia una volta arrivati a 160.
RepeatBehavior fa sì che l'animazione si ripeta all'infinito.
Come si può notare anche in questo non è necessario ricorrere a codice che intercetti il pulsante o che avvii l'animazione, perché è tutto gestito attraverso semplici proprietà.

Daniele Bochicchio, Cristian Civera

GLI AUTORI

Daniele Bochicchio è il content manager di ASPItalia.com, community che si occupa di ASP.NET, Classic ASP e Windows Server System. Il suo lavoro è principalmente di consulenza e formazione, specie su ASP.NET, e scrive per diverse riviste e siti. È Microsoft ASP.NET MVP, un riconoscimento per il suo impegno a supporto delle community e per l'esperienza maturata negli anni. Il suo blog è all'indirizzo http://blogs.aspitalia.com/daniele/

Cristian Civera è il content manager di WinFXItalia.com, una nuova community che è rivolta esclusivamente agli sviluppatori che intendono trarre il massimo vantaggio da WinFX. Cristian è Microsoft ASP.NET MVP ed utilizza fin dalle prime alpha private XAML ed Avalon, ora WPF. Il suo blog è all'indirizzo http://blogs.aspitalia.com/ricciolo/

QUERY UNIVERSALI CON FILTRI SPECIALI

LE APPLICAZIONI GESTIONALI SONO PIENE DI MASCHERE CHE PERMETTONO AGLI UTENTI DI EFFETTUARE INTERROGAZIONI, MA LA REALIZZAZIONE DI TALI INTERFACCE È SPESSO ONEROSA E RIPETITIVA. VEDIAMO COME SEMPLIFICARE IL PROBLEMA IN .NET

Capita sempre più spesso, nella scrittura di applicazioni gestionali, di dover passare moltissimo tempo, dopo la messa in produzione, a scrivere piccole modifiche per recepire questa o quella richiesta del cliente per ottenere una stampa particolare o un filtro eccentrico sui dati, magari non previsto in fase di analisi. E questo implica un continuo rimaneggiamento del codice.
In .NET, così come in tutti i linguaggi e gli ambienti di programmazione moderni non orientati specificatamente alla scrittura di gestionali (si legga Powerbuilder), per la soluzione di questo annoso problema, non volendo arrivare a soluzioni specifiche quali gli ipercubi, si procede con la normale scrittura di una maschera che, in qualche modo e con una serie di indicazioni dell'utente, arriva a generare la stringa con la query SQL desiderata. In genere si scrive parecchio codice, a meno che non si desideri una soluzione completamente in locale e cioè realizzata recuperando tutti i dati in griglia e affidandosi a questa per filtrare i dati. Proviamo a trovare una soluzione alternativa.

UN TEMPLATE PER LE QUERY DI SELEZIONE

Le query di selezione per le maschere utente di ricerca sono solitamente costituite da una parte invariante, cioè la tabelle o le tabelle che contengono il set di informazioni richieste, e una parte variabile che insiste soprattutto sui filtri di selezione, normalmente composti con una o più query. Semplificando con un esempio: si supponga di voler ottenere l'elenco degli ordini presenti nella tabella *Orders* di Northwind, il database di esempio di SQL Server 2000. Ecco un esempio di un'ipotetica query base con tutti i campi richiesti scritta in T-SQL di SQL Server 2000:

```
SELECT DISTINCT ORDERS.*, PRODUCTS.UNITPRICE,
                        SHIPPERS.COMPANYNAME
FROM  ORDERS INNER JOIN
    CUSTOMERS ON ORDERS.CUSTOMERID =
            CUSTOMERS.CUSTOMERID INNER JOIN
    [ORDER DETAILS] ON ORDERS.ORDERID =
          [ORDER DETAILS].ORDERID INNER JOIN
    PRODUCTS ON [ORDER DETAILS].PRODUCTID =
              PRODUCTS.PRODUCTID INNER JOIN
    CATEGORIES ON PRODUCTS.CATEGORYID =
            CATEGORIES.CATEGORYID INNER JOIN
    EMPLOYEES ON ORDERS.EMPLOYEEID =
        EMPLOYEES.EMPLOYEEID LEFT OUTER JOIN
    EMPLOYEETERRITORIES ON
                    EMPLOYEES.EMPLOYEEID =
     EMPLOYEETERRITORIES.EMPLOYEEID  INNER JOIN
    SHIPPERS ON ORDERS.SHIPVIA =
               SHIPPERS.SHIPPERID INNER JOIN
    SUPPLIERS ON PRODUCTS.SUPPLIERID =
              SUPPLIERS.SUPPLIERID INNER JOIN
    TERRITORIES ON
          EMPLOYEETERRITORIES.TERRITORYID =
          TERRITORIES.TERRITORYID INNER JOIN
    REGION ON TERRITORIES.REGIONID =
                          REGION.REGIONID
WHERE PRODUCTS.UNITSINSTOCK > 1
ORDER BY ORDERS.EMPLOYEEID
```

Essa contiene tutto il necessario in termini di sviluppo orizzontale, cioè di campi restituiti, ma è troppo poco filtrante perché restituisce praticamente tutti gli ordini a meno di un'unica condizione nella *where*. Ma se il nostro utente volesse effettuare selezioni più filtranti ci ritroveremmo col problema di dover scrivere query ad hoc o, peggio, di doverci far restituire comunque tutti i dati dal database filtrando applicativamente solo i dati richiesti dall'utente con uno spreco di tempo e di risorse notevoli.
Ed ecco una possibile soluzione; innanzitutto procediamo col delimitare le parti invarianti della query ed in particolare la testata:

```
<HEADER>
SELECT DISTINCT ORDERS.*,
    PRODUCTS.UNITPRICE, SHIPPERS.COMPANYNAME
FROM  ORDERS INNER JOIN
      CUSTOMERS ON ORDERS.CUSTOMERID =
              CUSTOMERS.CUSTOMERID INNER JOIN
      [ORDER DETAILS] ON ORDERS.ORDERID =
              [ORDER DETAILS].ORDERID INNER JOIN
      PRODUCTS ON [ORDER DETAILS].PRODUCTID =
              PRODUCTS.PRODUCTID INNER JOIN
      CATEGORIES ON PRODUCTS.CATEGORYID =
              CATEGORIES.CATEGORYID INNER JOIN
      EMPLOYEES ON ORDERS.EMPLOYEEID =
              EMPLOYEES.EMPLOYEEID LEFT OUTER JOIN
      EMPLOYEETERRITORIES ON
              EMPLOYEES.EMPLOYEEID =
        EMPLOYEETERRITORIES.EMPLOYEEID INNER JOIN
      SHIPPERS ON ORDERS.SHIPVIA =
              SHIPPERS.SHIPPERID INNER JOIN
      SUPPLIERS ON PRODUCTS.SUPPLIERID =
              SUPPLIERS.SUPPLIERID INNER JOIN
      TERRITORIES ON
              EMPLOYEETERRITORIES.TERRITORYID =
              TERRITORIES.TERRITORYID INNER JOIN
      REGION ON TERRITORIES.REGIONID =
              REGION.REGIONID
WHERE PRODUCTS.QUANTITYPERUNIT > 1
</HEADER>
```

E la parte di ordinamento che si trova come chiosa della select:

```
<FOOTER>
ORDER BY ORDERS.EMPLOYEEID
</FOOTER>
```

Volendo agire sulla query, un esempio relativamente semplice di filtro ulteriore potrebbe essere il seguente. Osserviamone solo la nuova parte *where* dando per scontata la parte iniziale della query che resta invariata, compresa la prima condizione di *where* e la clausola *order by finale:*

```
//FILTRO A
AND UPPER(CUSTOMERS.COMPANYNAME) LIKE
              UPPER('<VALORE>%')
```

Con il tag <*VALORE*> vengono indicati i valori di filtro immessi dall'utente. In questo caso si è supposto che l'utente voglia filtrare gli ordini solo di clienti che iniziano per un certo nome. Oppure ancora si supponga un filtro sul valore di sconto applicato sugli ordini:

```
//FILTRO B
AND [ORDER DETAILS].DISCOUNT >= <VALORE>
```

E, perché no, una condizione sulla data di spedizione:

```
//FILTRO C
AND ORDERS.SHIPPEDDATE >= cast('<VALORE>'
                              as datetime)
```

A parte i semplici filtri scalari, potrebbe nascere l'esigenza di applicare filtri più complessi come quelli che fanno uso di sub query, come nel caso seguente in cui il filtro in cui vengono selezionati solo gli ordini aventi una determinato spedizioniere:

```
//FILTRO D
AND ORDERS.SHIPVIA =
    (SELECT SHIPPERID
     FROM SHIPPERS
     WHERE SHIPPERID = <VALORE>)
```

In realtà l'esempio precedente si sarebbe potuto realizzare anche con una semplice condizione piatta, ma a scopo esemplificativo del tipo di filtro possibile si è introdotta questa sintassi più complessa. Infine si potrebbe porre la necessità di effettuare filtri con valutazioni statistiche, come nel seguente esempio che recupera solo gli ordini contenenti righe con prodotti il cui prezzo medio è minore di quello indicato dall'utente :

```
//FILTRO E
AND <VALORE> >
    (SELECT AVG(UNITPRICE)
     FROM PRODUCTS
     WHERE PRODUCTID = [ORDER
                   DETAILS].PRODUCTID)
```

È altresì vero che l'utente potrebbe voler scegliere una sola delle condizioni di filtro indicate o abbinarne più di una, pertanto la nostra *select* deve tener conto di tutte le possibilità.
Ma osserviamo meglio la struttura della parte variabile dei filtri *A* e *B*, cioè quella contenente il tag <*VALORE*>*:*

A) *AND UPPER(CUSTOMERS.COMPANYNAME) LIKE UPPER('<VALORE>%')*
B) *AND [ORDER DETAILS].DISCOUNT >= <VALORE>*

Fondamentalmente esse sono strutturate nella forma:

```
<NOME_CAMPO> <OPERATORE_DI_CONFRONTO>
                              <VALORE>
```

Il nome del campo rappresenta proprio il

I TUOI APPUNTI

nome del campo di tabella su cui effettuare il filtro, magari corredato dal prefisso di tabella (es. *CUSTOMERS. COMPANYNAME* o *[ORDER DETAILS].DISCOUNT*). L'operatore, invece, in SQL può assumere i segni >, <, >=, <=, *LIKE* o *NOT LIKE*. Infine il valore è il dato su cui effettuare il confronto; l'unico accorgimento da usare in questo campo è quello di decorare le stringhe con apici singoli. Pertanto, le valutazioni *A* e *B* sono di fatto racchiudibili in un'unica soluzione basata su uno dei solito tag:

```
<GENERIC_FILTER>
AND {FIELD} {OP} {VALUE}
<GENERIC_FILTER>
```

Se, invece, ci concentriamo sul filtro *C*, possiamo sintetizzarlo con la seguente sintassi:

```
<DATE_FILTER>
AND {FIELD} {OP} CAST('{VALUE}' as datetime)
</DATE_FILTER>
```

In pratica la parte del valore è inclusa nella funzione di *CAST* delle date. È da tener presente che questa sintassi è differente da motore a motore; nello specifico abbiamo usato la sintassi T-SQL di Microsoft SQL Server. Infine, per quanto riguardo i filtri *D* ed *E* possiamo procedere con la seguente sintassi:

```
<SHIPVIA>
AND ORDERS.SHIPVIA =
      (SELECT SHIPPERID
       FROM SHIPPERS
       WHERE SHIPPERID = ORDERS.SHIPVIA
               AND {FIELD} {OP} {VALUE})
</SHIPVIA>

<AVERAGE_PRICE>
AND {VALUE} {OP}
      (SELECT AVG(UNITPRICE)
       FROM PRODUCTS
       WHERE PRODUCTID = [
                         ORDER DETAILS].PRODUCTID)
</AVERAGE_PRICE>
```

Ricapitolando, osserviamo la nostra query di esempio strutturata con tag:

```
<HEADER>
SELECT DISTINCT ORDERS.*, PRODUCTS.UNITPRICE,
                          SHIPPERS.COMPANYNAME
FROM  ORDERS INNER JOIN
      CUSTOMERS ON ORDERS.CUSTOMERID =
              CUSTOMERS.CUSTOMERID INNER JOIN
      [ORDER DETAILS] ON ORDERS.ORDERID =
            [ORDER DETAILS].ORDERID INNER JOIN
      PRODUCTS ON [ORDER DETAILS].PRODUCTID =
                 PRODUCTS.PRODUCTID INNER JOIN
      CATEGORIES ON PRODUCTS.CATEGORYID =
               CATEGORIES.CATEGORYID INNER JOIN
      EMPLOYEES ON ORDERS.EMPLOYEEID =
           EMPLOYEES.EMPLOYEEID LEFT OUTER JOIN
      EMPLOYEETERRITORIES ON
                       EMPLOYEES.EMPLOYEEID =
        EMPLOYEETERRITORIES.EMPLOYEEID  INNER JOIN
      SHIPPERS ON ORDERS.SHIPVIA =
                   SHIPPERS.SHIPPERID INNER JOIN
      SUPPLIERS ON PRODUCTS.SUPPLIERID =
                 SUPPLIERS.SUPPLIERID INNER JOIN
      TERRITORIES ON
            EMPLOYEETERRITORIES.TERRITORYID =
            TERRITORIES.TERRITORYID INNER JOIN
      REGION ON TERRITORIES.REGIONID =
                               REGION.REGIONID
WHERE PRODUCTS.QUANTITYPERUNIT > 1
</HEADER>

<GENERIC_FILTER>
AND {FIELD} {OP} {VALUE}
</GENERIC_FILTER>

<DATE_FILTER>
AND ORDERS.{FIELD} {OP} CAST('{VALUE}'
                                  as datetime)
</DATE_FILTER>

<SHIPVIA>
AND ORDERS.SHIPVIA =
      (SELECT SHIPPERID
       FROM SHIPPERS
       WHERE SHIPPERID = ORDERS.SHIPVIA
               AND {FIELD} {OP} {VALUE})
</SHIPVIA>

<AVERAGE_PRICE>
AND {VALUE} {OP}
      (SELECT AVG(UNITPRICE)
       FROM PRODUCTS
       WHERE PRODUCTID = [ORDER
                          DETAILS].PRODUCTID)
</AVERAGE_PRICE>

<FOOTER>
ORDER BY ORDERS.EMPLOYEEID
</FOOTER>
```

Si può osservare come le uniche parti invarianti siano il tag *<HEADER>* e *<FOOTER>*; tutte le altre parti, invece, sono opzionali e possono addirittura ripetersi più volte, pertanto potrebbe porsi il caso in cui l'utente opti di filtrare per due diversi campi del tag *<GENERIC_FILTER>*, un solo campo di *<AVERAGE_ PRICE>* e nessun altro campo di filtro.

Ecco un esempio:

```
--parte invariante
WHERE PRODUCTS.UNITSINSTOCK > 1
--primo filtro di tipo <GENERIC_FILTER>
AND UPPER(CUSTOMERS.COMPANYNAME) LIKE
                                UPPER('AL%')
--secondo filtro di tipo <GENERIC_FILTER>
AND ORDERS.SHIPPEDDATE >= cast('19971205' as
                                   datetime)
--filtro di tipo <AVERAGE_PRICE>
AND 35 >
    (SELECT AVG(UNITPRICE)
     FROM PRODUCTS
     WHERE PRODUCTID = [ORDER
                        DETAILS].PRODUCTID)
```

Come fare per realizzare questo modello in modo automatico e che, magari, ci consenta di recepire le richieste di filtro dell'utente e di produrre dell'utente e di produrre una query SQL completa in modo automatico? Dopo aver introdotto la sintassi per scrivere query template, passiamo alla definizione di un nuovo elemento: il modello ad oggetti in

Fig. 1: *Una griglia di filtro in Microsoft Access 2003*

grado di rappresentare i filtri e cioè la parte dinamica della where delle nostre query template.

COSTRUZIONE DEL MODELLO AD OGGETTI DEI FILTRI

Accoppieremo al template che rappresenta la qery generica un dizionario di campi di filtro, cioè un elenco di campi, con relativo tipo, che l'utente può utilizzare nella composizione delle sue richieste di filtro sui dati.
Supponiamo che egli possa disporre di un dizionario di campi di filtro come mostrato nel riquadro. Fondamentalmente possiamo pensare la nostra *where* come una matrice a due dimensioni simile a quella che appare in Microsoft Access.
Sulle righe abbiamo le condizioni di *OR* e sulle colonne abbiamo le varie condizioni di *AND*; ciascuna cella rappresenta la singola condizione del tipo *CAMPO = VALORE* e che corrisponde ad un singolo filtro *<GENERIC_FILTER>*, *<SHIPVIA>*, ecc... La tabella può avere un numero indefinito di righe e di colonne, inoltre è possibile legare tra loro due o più di righe in una condizione di *AND* al posto della *OR* di default, in modo da ottenere virtualmente una sola riga con tutte le condizioni della prima riga in *AND* con quella della seconda riga, della terza, ecc...
Nel riquadro è possibile osservare un esempio di questo filtro: ritroviamo la tabella che abbiamo appena descritto e in ciascuna cella ritroviamo proprio l'applicazione dei filtri come risultato dell'associazione tra i filtri definiti nella query template e il dizionario di filtri associato ad essa. Lo sviluppo dell'esempio porterebbe ad una parte *where* siffatta:

```
WHERE ORDERS.EmployeeID > 1 AND
         --PARTE INVARIANTE PROVENIENTE DAL TAG
                                        <HEADER>
(
-- RIGA No. 1
            (           --Colonna 1
            (ORDERS.RequiredDate >=
                    CAST('19981012' as datetime))
            AND
                        --Colonna 2
            (UPPER(RTRIM(ORDERS.ShipName)) like
                                 UPPER('%MAERSK'))
            )

-- RIGA No. 2
OR
            (    --Colonna 1
                 (UPPER(RTRIM(ORDERS.CustomerID)) =
                                      UPPER('134'))
            )

-- RIGA No. 3 in AND con la RIGA No. 4
OR
            (    --Colonna 1 – Riga No. 3
                 (ORDERS.SHIPVIA =
                 (SELECT SHIPPERID
                  FROM SHIPPERS
                  WHERE SHIPPERID = ORDERS.SHIPVIA
                  AND UPPER(RTRIM(CompanyName))
                         like UPPER('CODEBEHIND%'))
            )
            AND
            --Colonna 2 – Riga No. 3
            (UPPER(RTRIM(ORDERS.ShipName)) like
                                 UPPER('MAERSK%'))
            AND
            --Colonna 3 – Riga No. 3
            (10.3 <
            (SELECT AVG(UNITPRICE)
             FROM PRODUCTS
             WHERE PRODUCTID = [ORDER
                           DETAILS].PRODUCTID)
```

Nome tag	Nome campo	Tipo campo	Descrizione
GENERIC_FILTER	ORDERS.ShipName	FieldType.Text	Modalità di Trasporto
GENERIC_FILTER	ORDERS.CustomerID	FieldType.Text	Codice Cliente
GENERIC_FILTER	[ORDER DETAILS].UNITPRICE	FieldType.FloatingPoint	Prezzo unitario del prodotto
DATE_FILTER	RequiredDate	FieldType.DateTime	Data Richiesta
SHIPVIA	CompanyName	FieldType.Text	Nome Trasportatore
AVERAGE_PRICE	UNITPRICE	FieldType.FloatingPoint	Prezzo Medio

Tabella 1: Un esempio di dizionario di campi di filtro

```
            )
            AND
            --Colonna 1 – Riga No. 4
            (UPPER(RTRIM(ORDERS.ShipName)) like
                                UPPER('%NEUROGEST%'))
            )

-- RIGA No. 5
OR
            (           --Colonna 1
                        (45 >
                        (SELECT AVG(UNITPRICE)
                           FROM PRODUCTS
                           WHERE PRODUCTID = [ORDER
                                    DETAILS].PRODUCTID)
                        )
            )
)
```

Ci si può rendere conto della potenza e della versatilità di questo approccio: partendo da una query di template e da un dizionario di campi di filtro possiamo ottenere un numero infinito di combinazioni senza ogni volta riscrivere la query.

DALLA TEORIA ALLA PRATICA: IL MODELLO AD OGGETTI

Questa lunga premessa ci porta inevitabilmente alla realizzazione pratica di questa mini architettura di filtri. Banalmente dobbiamo disporre di una stringa nella quale conservare la query template. A questo punto passiamo al codice, che racchiuderemo nel namespace *platformBehind.DataManipulation.Filter*, e definiamo una serie di enumerativi; per gli operatori SQL di confronto dei valori:

```
[Serializable]
public enum MathOperator
{
   Equal = 0,         \\operatore =
   Greater = 1,       \\operatore >
   GreaterOrEqual = 2,          \\operatore >=
   Lesser = 3,        \\operatore <
   LesserOrEqual = 4,           \\operatore <=
   LikeOp = 5,        \\operatore LIKE '%STRINGA%'
   LikeLeftOp = 6,  \\operatore LIKE '%STRINGA'
   LikeRigthOp = 7 \\operatore LIKE 'STRINGA%'
}
```

E un enumerativo per la gestione dei tipi di dati supportati dal nostro sistema di filtri:

```
[Serializable]
public enum FieldType
{
   Numeric = 1,
   Text = 2,
   DateTime = 3,
   FloatingPoint = 4
}
```

Introduciamo finalmente l'oggetto *Field*, cioè il singolo campo del dizionario dei filtri, dunque il mattone fondamentale della nostro sistema di filtri:

```
[Serializable]
public class Field
{
   private string mFieldName;
   private string mStructName;
   private FieldType mFieldType;
   private int mFieldId;
   private bool mSystem;
   private string mSpecialKey;

   public int FieldId //identificativo numerico univoco
                                                del campo
   {
        get { return mFieldId; }
        set { mFieldId = value; }
   }

   public string StructName //nome del tag del query
                             template nel quale è definito
   {
       get { return mStructName; }
       set { mStructName = value; }
   }

   public string FieldName //nome del campo fisico
                                               nella query
```

```
    {
        get { return mFieldName; }
        set { mFieldName = value; }
    }

    public FieldType FieldType //tipologia del campo
    {
        get {return mFieldType; }
        set {mFieldType = value; }
    }

}
```

Nella classe *Field* ritroviamo tutti gli elementi fin qui esposti: il nome del campo fisico, il nome del tag che ospita il campo (ad esempio *GENERIC_FILTER*) e la sua tipologia secondo l'enumerativo *FieldType* definito in precedenza. Non ci resta che racchiudere i campi di dizionario in una classe specifica di tipo contenitore, dopo aver meglio definito la classe *Field* attraverso una specializzazione:

```
//classe derivata da field che aggiunge proprietà di
                                    rappresentazione a video
public class UIField : Field
{
    private string mDescription;
    private int mPosition;

    //descrizione
    public string Description
    {
        get { return mDescription; }
        set { mDescription = value; }
    }

    //posizione del campo rispetto agli altri dello stesso
                                                  dizionario
    public int Position {
        get { return mPosition; }
        set { mPosition = value; }
    }
}

//classe contenitore del dizionario di filtri
[Serializable]
public class FilterDictionary : Hashtable
{
    //costruttore di default
    public FilterDictionary() : base() { }

    //costruttore di serializzazione
    public FilterDictionary(SerializationInfo info,
                         StreamingContext context) :
                                   base(info, context) { }

    //
```

```
    public new UIField this[object key] {
        get { return (UIField)base[key]; }
        set { base[key] = value; }
    }

    public void Add(object key, UIField value) {
        base.Add(key, value);
    }

    public bool ContainsValue(UIField value) {
        return base.ContainsValue(value);
    }
}
```

È interessante notare che tutte le classi e gli enumerativi recano l'attributo di serializzazione, comportamento che assume maggiore evidenza nella classe *FilterDictionary* che, a questo dettaglio, aggiunge il costruttore di serializzazione, accorgimento necessario se si intende serializzare una classe derivante da oggetti nel namespace *System.Collections*.
Il passaggio da un campo del dizionario dei filtri, ad una cella di filtro della nostra tabella di filtri utente è più semplice di quel che sembra, d'altro canto si tratta solo di aggiungere l'operatore SQL di confronto e il valore da confrontare, che è proprio quello che fa la classe *Filter*, derivando da *Field:*

```
[Serializable]
public class Filter : Field
{
    MathOperator mMathOperator;
    string mValue;

    public Filter() { }

    public Filter(string name, FieldType type, string
                                             value) {
        FieldName = name;
        FieldType = type;
        mValue = value;
    }

    public string Value
    {
        get {return mValue; }
        set {mValue = value; }
    }

    public MathOperator MathOperator
    {
        get { return mMathOperator; }
        set { mMathOperator = value; }
    }

}
```

Ricordando la struttura della nostra riga di filtro, con tutti i campi *Filter* in *AND* tra loro, definiamo una classe in grado di rappresentare questa informazione e cioè una semplice collezione di *Filter* in *AND* tra loro:

```
[Serializable]
public class ANDFilterCollection : Hashtable
{
   public ANDFilterCollection() : base() { }

   public ANDFilterCollection(SerializationInfo info,
                              StreamingContext context)
                              : base(info, context){ }
   public new Filter this[object key]
   {
      get { return (Filter)base[key]; }
      set { base[key] = value; }
   }

   public void Add(object key, Filter value)
   {
      base.Add(key, value);
   }

   public bool ContainsValue(Filter value)
   {
      return base.ContainsValue(value);
   }
}
```

Le righe, però, sono in *OR* tra loro, pertanto è necessario un oggetto contenitore di righe. Eccolo:

```
[Serializable]
public class UserFilter : Hashtable     {
   string mFilterName;
   int mFilterId;

   public override void GetObjectData(
      SerializationInfo info, StreamingContext context) {
      base.GetObjectData(info, context);
      if (mFilterName == null)
         mFilterName = string.Empty;
         info.AddValue("mFilterName", mFilterName,
                               mFilterName.GetType());
         info.AddValue("mFilterId", mFilterId,
                               mFilterId.GetType());
   }

   public UserFilter() : base() {
      mFilterId = this.GetHashCode();
   }

   public UserFilter(SerializationInfo info,
                           StreamingContext context)
                           : base(info, context) {
      mFilterId = this.GetHashCode();
      try {
         mFilterName =  "" + info.GetString(
                                       "mFilterName");
         mFilterId = info.GetInt32("mFilterId");
      }
      catch {
         //
      }
   }
```

No.	Operatore di riga	Filtro Colonna 1	Filtro Colonna 2	Filtro Colonna 3
1		**Struttura:** <GENERIC_FILTER> **Campo:** ORDERS.ShipName **Operatore:** LIKE **Valore:** '%MAERSK'	**Struttura:** <DATE_FILTER> **Campo:** RequiredDate **Operatore:** >= **Valore:** 12-10-1998	
2	*OR*	**Struttura:** <GENERIC_FILTER> **Campo:** ORDERS.CustomerID **Operatore:** = **Valore:** 134		
3	*OR*	**Struttura:** <SHIPVIA> **Campo:** CompanyName **Operatore:** LIKE **Valore:** 'CODEBEHIND%'	**Struttura:** <GENERIC_FILTER> **Campo:** ORDERS.ShipName **Operatore:** LIKE **Valore:** 'MAERSK%'	**Struttura:** <GENERIC_FILTER> **Campo:** [ORDER DETAILS].UNITPRICE **Operatore:** < **Valore:** 10.3
4	*AND*	**Struttura:** <GENERIC_FILTER> **Campo:** ORDERS.ShipName **Operatore:** LIKE **Valore:** '%NEUROGEST%'		
5	*OR*	**Struttura:** <AVERAGE_PRICE> **Campo:** UNITPRICE **Operatore:** > **Valore:** 45		

Tabella 2: Tabella logica di filtro (le colonne sulla stessa riga sono in AND tra loro)

```
public new ANDFilterCollection this[object key] {
    get { return (ANDFilterCollection)base[key]; }
    set { base[key] = value; }
}

public string FilterName {
    get {return mFilterName; }
    set {mFilterName = value; }
}

public int FilterId {
    get {return mFilterId; }
    set {mFilterId = value; }
}

public void Add(object key, ANDFilterCollection
                                        value) {
    base.Add(key, value);
}

public  bool ContainsValue(ANDFilterCollection
                                        value) {
    return base.ContainsValue(value);
}

}
```

IL MODELLO AD OGGETTI IN AZIONE

Siamo pronti a mettere alla prova il nostro nuovo modello ad oggetti. Cominciamo col compilare il dizionario dei filtri della nostra query template di esempio; ecco un esempio di come definire il campo *ORDERS.ShipName* definito nella struttura *GENERIC_FILTER:*

```
FilterDictionary dictionary = new FilterDictionary();

UIField field = new UIField();
field.FieldName = "ORDERS.ShipName";
field.FieldType = FieldType.Text;
field.StructName = "GENERIC_FILTER";
field.Position = 1;
field.Description = "Modalità di Trasporto";
dictionary.Add(field.StructName + "." +
                              field.FieldName, field);
```

Ecco, invece, come adoperare il modello ad oggetti per realizzare il filtro vero e proprio. Il codice è solo a titolo esemplificativo visto che, della nostra ipotetica matrice di filtri, viene creata solo una riga che contiene una sola colonna.

```
//creazione del filtro
UserFilter filter = new UserFilter();
Filter filterit;
UIField colitem;

//creazione di una riga
ANDFilterCollection ANDfilter = new
                               ANDFilterCollection();
filter.Add(Guid.NewGuid(), ANDfilter);

//creazione di una cella di filtro
filterit = new Filter();
filterit.StructName = "GENERIC_FILTER";
filterit.FieldName = "ORDERS.ShipName";
filterit.MathOperator = MathOperator.Equal;
filterit.FieldType = FieldType.Text;
filterit.Value = "MAERSK";
ANDfilter.Add(Guid.NewGuid(), filterit);
```

CONCLUSIONI

Abbiamo introdotto un sistema per realizzare filtri SQL dinamici basati su una query scritta in forma di template e quindi pensata non per soddisfare una singola richiesta ma una classe di richieste sullo stesso set di dati, in grado di effettuare confronti e selezioni anche molto complessi, perché basati anche su subquery correlate come dimostra l'esempio usato nell'articolo. Abbiamo inoltre introdotto uno schema logico basato su una matrice di condizioni in *AND* e in *OR* per rappresentare i filtri, e cioè le parti variabili della nostra query SQL ipotetica e abbiamo mappato tale schema su un modello ad oggetti C#.
Siamo a buon punto, perché ormai abbiamo la ricetta e tutti gli ingredienti, ma non abbiamo ancora l'elemento più importante: la frittata, cioè il risultato finale. Nel prossimo numero, infatti, vedremo come scrivere un motore analizzatore di testo che, a partire da una query template, da un dizionario di filtri (un oggetto *FilterDictionary*) e dal filtro vero e proprio (un oggetto *UserFilter*), sia in grado di produrre la query SQL vera e propria pronta ad essere eseguita dal nostro motore di accesso ai dati. Vedremo, inoltre, come scrivere query per SQL Server e per Oracle senza modificare la logica della query ne il modello ad oggetti, ma semplicemente agendo su due diverse query template. Come tocco finale, vedremo come realizzare una semplice griglia *WinForms* che consenta ad un utente di definire il proprio filtro in modo grafico ; griglia già pronta ad essere usata nelle nostre applicazioni e che necessita solo di essere inizializzata con un dizionario di filtri.

Vito Vessia

L'AUTORE

Vito Vessia progetta e sviluppa applicazioni e framework in .NET, COM(+) e Delphi occupandosi degli aspetti architetturali. Scrive da anni per le principali riviste italiane di programmazione ed è autore del libro

• *PROGRAMMARE IL CELLULARE* (Hoepli) 2002

sulla programmazione dei telefoni cellulari connessi al PC con protocollo standard AT+. Può essere contattato tramite e-mail all'indirizzo vvessia@katamail.com.

TRASFORMARE DI TUTTO CON XSL

IN QUESTO ARTICOLO CI OCCUPEREMO DI ALCUNI ASPETTI DI XSL ABBASTANZA SOFISTICATI. PARLEREMO DI ORDINAMENTO DEI DATI, DI GESTIONE DELLE QUERY INNESTATE E DI MODULARITÀ DEI FILE...

Per coloro che si sono persi *le puntate precedenti*: abbiamo indicato come XSL un linguaggio in grado di convertire un qualunque documento XML in un qualunque altro formato. Nella scorsa puntata invece ci siamo occupati di XPATH, un argomento strettamente collegato ad XSL, ovvero la tecnica che consente di estrarre da un documento XML un certo numero di nodi sulla base di una query di selezione. Una volta estratti i nodi è possibile trasformare solo questi, che rappresentano la porzione di dati desiderata, nel formato che si desidera tramite XSL. XPATH è abbastanza esteso e complesso da rappresentare un linguaggio a sé stante, un po' come SQL rappresenta un linguaggio per l'interrogazione dei dati, da utilizzare poi in linguaggi di più ampio respiro come Visual Basic, C#, C++, PHP etc.
In questo numero analizzeremo ancora XSL parlando di come ordinare i dati prima di applicare una trasformazione, e di qualche altra tecnica collaterale con la quale termineremo questa parte base del nostro corso XSL.

ORDINAMENTO DEI NODI

Come sappiamo l'elemento <xsl:for-each> consente di effettuare un ciclo sui nodi da trasformare. Complemento, dell'elemento <xsl:for-each> è l'elemento <xsl:sort>. Come suggerisce il nome <xsl:sort> si occupa di ordinare i nodi recuperati con <xsl:for-each>. Come sempre partiamo prima da un esempio per comprendere meglio.
Prendiamo in considerazione una sorgente di dati Xml una semplice lista di indirizzi :

```
<indirizzi>
    <indirizzo>
        <id>1</id>
        <nome>Antonio</nome>
        <cognome>Rossi</cognome>
        <via>G.Verdi, 3</via>
        <comune>Roma</comune>
    </indirizzo>

    <indirizzo>
        <id>2</id>
        <nome>Giuseppe</nome>
        <cognome>Bianchi</cognome>
        <via>G.Rossini, 4</via>
        <comune>Milano</comune>
    </indirizzo>
</indirizzi>
```

Poniamo il caso che, nella trasformazione, sia necessario ordinare prima i record in ordine alfabetico per cognome.
Applichiamo quindi la trasformazione utilizzando il seguente foglio di stile:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
        xmlns:xsl="http://www.w3.org/
            1999/XSL/Transform>
<xsl:template match="/">
    <xsl:for-each select="//indirizzo">
        <xsl:sort select="cognome"
            order="ascending"
            data-type="text"/>
        <div>
        <xsl:value-of select="nome"/>
         
        <xsl:value-ofselect="cognome"/>
        </div>
    </xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

La lettura, grazie anche a quanto abbiamo appreso di XPATH, a questo punto dovrebbe essere abbastanza agevole:

1. Per mezzo di un ciclo <xsl:for-each> andiamo a selezionare tutti (si noti la path "//")

i nodi che hanno nome "indirizzo" indipendentemente dalla loro posizione.

2. Immediatamente sotto <xsl:for-each> inseriamo un elemento <xsl:sort> per presentare i nodi secondo l'ordinamento definito dagli attributi: select (nodo o attributo da prendere in considerazione per l'ordinamento), order (tipo di ordinamento; ascendente o discendente) e data-type (tipo di dati; testo o numeri).

3. Inseriamo, all'interno del ciclo <xsl:for-each> e sotto <xsl:sort> la logica di trasformazione.

Il risultato della trasformazione sarà appunto :

```
Giuseppe Bianchi
Antonio Rossi
```

GLI ATTRIBUTI DELL'ELEMENTO <XSL:SORT>

Com'è intuibile la trasformazione viene impostata mediante gli attributi di <xsl:sort>.

select - È l'attributo più importante che definisce, in linguaggio XPATH, quale espressione utilizzare nel confronto per ordinare due nodi e stabilire la priorità. Indica in pratica dove risiedono i dati da ordinare: in un altro nodo (solitamente più interno), in un attributo , ecc...
Le possibilità di definire dove stanno i dati da utilizzare nel confronto sono praticamente infinite e sono date dalla flessibilità di XPATH, tenendo comunque in mente che l'espressione da scrivere nel select di <xsl:sort> è sempre relativa rispetto al nodo corrente di <xsl:for-each>.
Solo qualche esempio:

<xsl:sort select="../@categoria"/>
usa il valore dell'attributo "categoria" del nodo superiore rispetto a quello corrente

<xsl:sort select="@id"/>
usa il valore dell'attributo "id" del nodo corrente

<xsl:sort select="nome/@id"/>
usa il valore dell'attributo "id" del nodo "nome" contenuto in quello corrente

<xsl:sort select="substring(cognome,1,3)"/>
usa solo le prime 3 lettere del valore del nodo "cognome" contenuto in quello corrente

order - definisce l'ordine da applicare alla lista dei nodi e ammette solo i valori: ascending (ascendente) o descending (discendente).

data-type - indica il tipo di dati da ordinare e supporta soltanto i valori : text (ordinamento alfanumerico) e number (ordinamento numerico).

Gli elementi <xsl:attribute> e <xsl:attribute-set>
L'elemento <xsl:attribute> serve per creare un attributo da inserire nell'elemento di output corrente.
Ad esempio questo pezzo codice :

```
...
<table>
<xsl:attribute name="border">0</xsl:attribute>
<xsl:attribute
                  name="cellpadding">0</xsl:attribute>
<xsl:attribute name="width">100%</xsl:attribute>
<tr>
            <td></td>
</tr>
</table>
...
```

PUNTATE PRECEDENTI

Come già detto XSL è un linguaggio per la trasformazione dei dati. Supponiamo che disponiate di un file XML così composto:

```
<?xml version="1.0"?>
<helloworld>
     <titolo>Hello World</titolo>
    <messaggio>
             Benvenuti su ioProgrammo
    </messaggio>
</helloworld>
```

E il corrispondente file di trasformazione XSL

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:
xsl="http://www.w3.org/1999/XSL/Tra
nsform">
<xsl: template match="/">
<html>
<head>
<title>Esempio  Hello</title>
</head>
<body>
<h1><xsl:value-of select="hel-
loworld/titolo"/></h1>
</body></html>
</xsl:template>
</xsl:stylesheet>
```

Potete effettuare la trasformazione aggiungendo le seguenti linee all'inizio del file XML.

```
<?xml version="1.0"?>
<?xml-stylesheet type=
                         "text/xsl" href=
                         "nomefile.xsl"?>
```

Oppure usando JavaScript:

```
<SCRIPT language =
                         "javascript">
  function transform(){
     var srcTree = new
                 ActiveXObject("Msxml2
                  .DOMDocument.4.0");
     srcTree.async=false;
     srcTree.load(
                     "helloworld.xml");
     var xsltTree= new
                 ActiveXObject("Msxml2
                 .DOMDOCUMENT.4.0");
     xsltTree.async = false;
     xsltTree.load(
                     "helloworld.xsl");

     return srcTree
                      .transformNode(
                           xsltTree);
  }
</SCRIPT>
```

Produrrà, come output il seguente :

```
<table border="0" cellpadding="0" width="100%">
    <tr>
        <td>
        </td>
    </tr>
</table>
```

Notiamo quindi come l'elemento <xsl:attribute> definisca l'attributo contenuto in name all'interno dell'elemento di output nel quale è inserito.
La cosa interessante è che gli attributi si possono raccogliere in "collezioni" mediante l'attributo <xsl:attribute-set> da inserire sotto il nodo radice del foglio di stile.
Nel caso precedente avremmo quindi potuto raccogliere tutti gli attributi in un elemento <xsl:attribute-set> in questo modo:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                        xmlns:xsl="http://www.w3.org
                                /1999/XSL/Transform>
<xsl:attribute-set name="table-def">
<xsl:attribute name="border">0</xsl:attribute>
<xsl:attribute name="cellpadding">0</xsl:attribute>
<xsl:attribute name="width">100%</xsl:attribute>
</xsl:attribute-set>
...
</xsl:stylesheet>
```

In questo modo nel codice di output sarebbe stato sufficiente inserire il nome attribuito al set di attributi in questo modo:

```
...
<table xsl:use-attribute-sets="table-def">
        <tr>
                <td></td>
        </tr>
</table>
...
```

xsl:use-attribute-sets (unico attributo definito nello schema XSL) permette infatti di connettere ad un elemento di output una lista di attributi definita con nome.
In questo modo è possibile definire più liste di attributi per i nodi da impiegare in parti diverse del codice di output migliorandone gestione e leggibilità.

Utilizza questo spazio per le tue annotazioni

INCLUSIONE DI PARTI DI CODICE XSL

L'inclusione di parti di codice XSL assomiglia agli include presenti in ASP o PHP e consente di dividere il codice XSL in files separati.
Questa pratica presenta due grossi vantaggi :

1. riutilizzo delle stesse funzioni in fogli di stile diversi
2. miglioramento della gestione e della leggibilità del codice

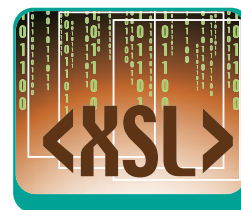
È possibile ad esempio mettere in un foglio di stile separato alcuni parametri (colore, caratteri ecc...) utilizzati nella trasformazione in modo da poterli cambiare in un unico punto anziché andarli a ricercare in tutti i fogli di stile.
L'inclusione di parti di codice può essere effettuata tramite due elementi <xsl:import> e <xsl:include>.
Entrambi hanno la stessa sintassi con un solo attributo href costituito dall'url relativo o assoluto del secondo foglio di stile da includere.
La differenza fondamentale tra <xsl:import> e <xsl:include> è che nel primo tutte le regole e i template definiti nel foglio di stile importato hanno la precedenza (nel caso di nome uguale) rispetto a quelli del foglio di stile corrente. Per questo <xsl:import> deve essere inserito come primo elemento sotto il nodo radice <xsl:stylesheet> del foglio di stile mentre per <xsl:include> non è necessario rispettare tale ordine.
Per semplicità noi utilizzeremo come esempio l'elemento <xsl:include>.
Passiamo quindi a vedere nella pratica l'utilizzo di un <xsl:include>.
Come sorgente XML utilizziamo la lista di indirizzi che abbiamo visto in precedenza.
Impostiamo quindi il foglio di stile da includere che metteremo nella stessa directory del foglio di stile principale e chiameremo common.xsl per far immediatamente comprendere che si tratta di codice che può essere utilizzato da più fogli di stile.
Inseriamo in common.xsl solo una collezione di attributi applicabili ad una tabella HTML:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                                                >
<xsl:attribute-set name="table-def">
<xsl:attribute name="border">0</xsl:attribute>
<xsl:attribute
                name="cellpadding">0</xsl:attribute>
<xsl:attribute name="width">100%</xsl:attribute>
</xsl:attribute-set>
</xsl:stylesheet>
```

Passiamo quindi a creare il foglio di stile principale da applicare alla sorgente XML definendo al suo interno un riferimento <xsl:include> a common.xsl e utilizzando, a questo punto, la colle-

zione di attributi come se fosse stata definita all'interno del foglio principale:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                    xmlns:xsl="http://www.w3.org
                          /1999/XSL/Transform>
<xsl:output method="html"/>
<xsl:include href="Common.xsl"/>
<xsl:template match="/">
<table xsl:use-attribute-sets="table-def">
<xsl:for-each select="//indirizzo">
<tr>
        <td><xsl:value-of select="nome"/></td>
      <td><xsl:value-of select="cognome"/></td>
        <td><xsl:value-of select="via"/></td>
      <td><xsl:value-of select="comune"/></td>
</tr>
</xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

Ai veterani dei linguaggi di scripting per il web come ASP o PHP non sarà sfuggita l'utilità della possibilità di inclusione per garantire la modularità dei fogli di stile.

RIFERIMENTO A SORGENTI XML ESTERNE

Ugualmente preziosa, anche se per altri motivi, è la possibilità di far riferimento a sorgenti XML esterne rispetto al documento che stiamo trasformando.
Il riferimento a sorgenti esterni si realizza attraverso una funzione del tutto simile alle funzioni XPATH che abbiamo visto la puntata precedente : la funzione document().

LA FUNZIONE DOCUMENT()

La funzione document() è molto versatile perché restituisce oggetti diversi a seconda del

ORDINAMENTO DI DATE

I tipi di dati che prende in considerazione l'algoritmo di <xsl:sort> sono soltanto text e number, pertanto ci si può trovare in difficoltà di fronte alla necessità di dover ordinare dei nodi per data. Si supponga ad esempio di dover ordinare per data discendente il seguente file Xml contenente dei logs:

```
<?xml version="1.0"?>
<logs>
        <log date="31/12/2005">Programma Avviato</log>
        <log date="6/1/2006">Programma Terminato</log>
        <log date="31/12/2006">Programma Avviato</log>
</logs>
```

Essendo omogenea la forma di presentazione della data (giorno/mese/anno) la soluzione sta nello "spezzettare" la stringa dell'attributo date con le funzioni di manipolazione di stringa offerte da XPATH. Così che:
ß substring-before(@date,'/') - individua il giorno (la sottostringa prima del primo "/")
• substring-before(substring-after(@date,'/'),'/') - individua il mese (la sottostringa prima del secondo "/")
• substring-after(substring-after(@date,'/'),'/') - individua l'anno (la sottostringa dopo del secondo "/")
Si applicano quindi le tre espressioni a tre differenti <xsl:sort> avendo cura ovviamente di mettere prima l'espressione che estrae l'anno, poi il mese ed infine il giorno:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/199
                                          /XSL/Transform">
<xsl:template match="/">
        <xsl:for-each select="//log">
<xsl:sortselect="substringafter (substringafter(@date,'/'),'/')"
data type="number"order= "descending"/>
<xsl:sort select="substring-before (substring-after(@date,'/'),'/')"
                              data-type="number"
                              order="descending"/>
<xsl:sort select="substring-before (@date,'/')"
                                 data-type="number"
                                order="descending"/>
<div>
<xsl:value-of select="@date"/> - <xsl:value-of select="."/>
</div>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Ottenendo come risultato della trasformazione, correttamente :
31/12/2006 - Programma Terminato
6/1/2006 - Programma Terminato
31/12/2005 - Programma Avviato
Si noti che il confronto, nell'ambito della sottostringa, è stato impostato come numerico (data-type="number) . L'algoritmo di sorting andrà infatti a confrontare in ordine discendente numerico prima la porzione dell'anno, quindi del mese ed infine del giorno.
Lo stesso risultato potremmo in realtà ottenerlo ricorrendo alle "funzioni estese" offerte dai più avanzati engine di trasformazione (.NET, php, java ecc...) tuttavia questa soluzione consente di non legarsi troppo ad una piattaforma di trasformazione per una funzione tutto sommato banale.

numero di parametri utilizzati:

1. document('doc.xml') - Utilizzando un solo parametro questo viene inteso come URL e l'oggetto restituito sarà il documento XML corrispondente come set di nodi.
2. document('doc.xml',//indirizzo) - Utilizzando due parametri il primo viene inteso come URL ed il secondo come il set di nodi a cui fare riferimento e viene restituito quest'ultimo.
3. document() - Utilizzando la funzione senza parametri viene restituito il foglio di stile stesso come documento XML.

Ma come utilizzare in pratica questa possibilità?
Poniamo di avere il nostro solito file XML degli indirizzi ma di aver attribuito ad ogni indirizzo una categoria sotto forma di codice :

```
<?xml version="1.0"?>
<indirizzi>
        <indirizzo>
                <id>1</id>
                <nome>Antonio</nome>
                <cognome>Rossi</cognome>
                <via>G.Verdi, 3</via>
                <comune>Roma</comune>
                <categoria>2</categoria>
        </indirizzo>
        <indirizzo>
                <id>2</id>
                <nome>Giuseppe</nome>
                <cognome>Bianchi</cognome>
                <via>G.Rossini, 4</via>
                <comune>Milano</comune>
                <categoria>1</categoria>
        </indirizzo>
</indirizzi>
```

Mettiamo quindi di aver definito le categorie in un secondo file XML (categorie.xml) :

```
<?xml version="1.0"?>
<categorie>
        <categoria cod="1">Lavoro</categoria>
        <categoria cod="2">Privato</categoria>
</categorie>
```

Come facciamo a collegare le descrizioni contenute nel secondo file ai nominativi contenuti nel primo file? Risolviamo il problema con questo foglio di stile:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0"
                    xmlns:xsl="http://www.w3.org
                    /1999/XSL/Transform>
<xsl:param name="categorie"
  select="document('categorie.xml')"></xsl:param>
<xsl:template match="/">
<table>
      <xsl:for-each select="//indirizzo">
        <tr>
          <td><xsl:value-of select="nome"/></td>
          <td>
          <xsl:value-of select="cognome"/></td>
          <td><xsl:value-of select="via"/></td>
          <td><xsl:value-of select="comune"/></td>
          <td>
          <xsl:value-of select=
                    "$categorie//categoria[@cod=
                              current()/categoria]"/>
          </td>
      </tr>
    </xsl:for-each>
</table>
</xsl:template>
</xsl:stylesheet>
```

In pratica assegniamo il node-set prodotto dalla lettura del secondo file XML ad un parametro "categorie" che viene poi utilizzato nel ciclo <xsl:for-each>. Quando si tratta di estrarre il valore della categoria relativa all'indirizzo usiamo la sintassi XPATH per trovare nel node-set il nodo categoria che abbia l'attributo cod corrispondente al valore del nodo "categoria" posto sotto al nodo "indirizzo" corrente.
Da notare l'utilizzo della funzione XSL current() che sta ad indicare il nodo corrente della funzione <xsl:for-each> che serve per evitare ambiguità nel riferimento ai nodi.

CONCLUSIONI

Con questa puntata termina la panoramica sull'XSL in generale. Il linguaggio offre anche una serie di altre funzioni "minori" e un'infinita gamma di "trucchi" a disposizione del programmatore. C'è da dire che altre funzioni, molto interessanti, sono offerte dalla possibilità di estensione al linguaggio dalle varie piattaforme di implementazione (.NET, PHP, Java ecc...). In alcune applicazioni (Web ma non solo) XSL costituisce uno dei fondamenti per costruire dei framework personali anche molto ricchi e complessi. Spesso è una delle soluzioni più indicate per effettuare trasformazioni rapide da un formato all'altro utilizzando XML per lo scopo principale per cui è nato, ovvero per essere un linguaggio intermedio e di trasporto.

Francesco Smelzo

SOFTWARE SUL CD

ANKHSETUP 0.5.5

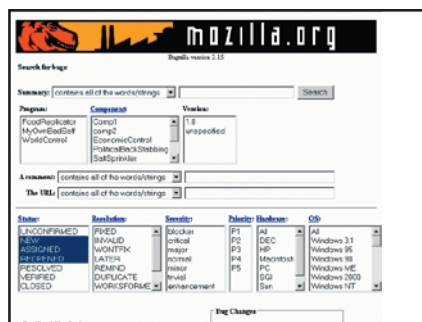
PER UTILIZZARE SUBVERSION DA VISUAL STUDIO.NET

SubVersion è uno strumento di controllo della revisione che si sta affermando molto rapidamente nel mondo della programmazione. Consente di tenere sotto controllo le modifiche effettuate al codice generando una serie di log che consentono di capire quando e come determinate parti dell'applicazione siano state modificate. Eventualmente è possibile tornare a versioni precedenti oppure generare il changelog automaticamente. Questo Add In consente ai programmatori .NET di utilizzare SubVersion direttamente dai proprio progetti e dall'ambiente di programmazione Visual Studio. Subversion sta lentamente soppiantando il fratello maggiore CVS che nel tempo aveva conquistato larghe fette di mercato. Alcuni progetti molto attivi come ad esempio il kernel di Linux sono passati a subversion

Directory: /AnchSetup

BUGZILLA 2.20

PER TENERE SOTTO CONTROLLO I BACHI DEL SOFTWARE

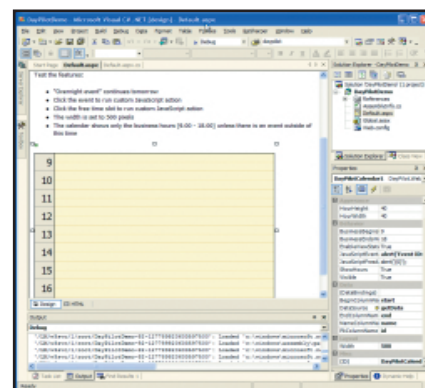
Un buon software evolve nel tempo sulla base delle indicazioni dei suoi utenti. E tutti i software nascono con qualche imperfezione che solitamente viene fuori proprio in fase di utilizzo intensivo dell'applicazione da parte di un considerevole numero di problemi.

Ma come tener traccia delle segnalazioni effettuate dagli utenti? Bugzilla è una Web Application che consente di memorizzare i bug, ordinarli secondo precisi ticket e così garantisce al programmatore di poter intervenire in modo organico, sistemando i bachi, elaborando patch, eventualmente segnalando all'utente falsi bug.

Un'applicazione senza dubbio utile che risolve uno dei maggiori problemi del ciclo di sviluppo. Attualmente bugzilla è lo strumento più diffuso per il controllo dei bug sui progetti opensource e molto spesso anche in caso di progetti commerciali

Directory: /Bugzilla

DAYPILOT 1.0.3

UN CONTROLLO ASP.NET PER LA GESTIONE DEGLI APPUNTAMENTI

Intelligente questo controllo! Consente di creare un'applicazione ASP .NET che offre funzionalità del tutto simili a quelle utilizzate da Outlook nella sua parte relativa alla gestione degli impegni.

Il controllo è molto solido e ben strutturato e offre una serie di funzioni che risultano piuttosto comode per un programmatore, molto più evolute di un normale calendario

Directory: / daypilot

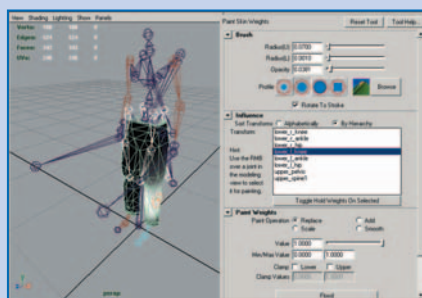
ANIMADEAD 2.0

PER CREARE ANIMAZIONI PARTENDO DA UNO SCHELETRO

Una libreria che sfrutta il concetto di skeletal animation.

Si inizia disegnando con un qualunque ambiente 3D lo scheletro di un soggetto, su questo scheletro si costruiscono i movimenti che saranno pilotati dalla libreria.

Si tratta di un progetto per alcuni versi ancora embrionale, ma che lascia intravedere un approccio piuttosto innovativo all'animazione di modelli tridimensionali

Directory: / Animadead

ECLIPSE 3.1.1

L'IDE MODULARE PER OGNI ESIGENZA

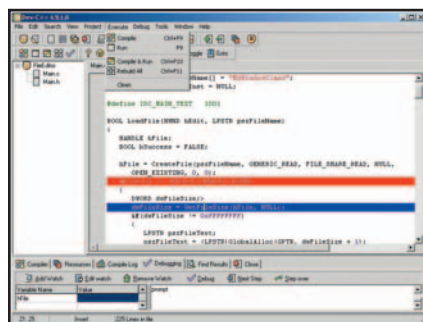
Eclipse è ormai un contenitore di immense dimensioni. L'IDE fa da ponte di raccordo, fra diverse tecnologie e linguaggi.

È possibile utilizzarlo per scrivere applicazioni Java, ma anche estenderlo per mezzo di plugin e renderlo compatibile con PHP, con Python e con una serie incredibile di altri linguaggi. Si tratta ormai di un progetto che ha assunto proporzioni colossali e al quale stanno via via aderendo una serie di software house dai nomi altisonanti quali IBM e persino Macromedia. Un'indispensabile dunque che si propone ormai come cuore pulsante dell'intero mondo dello sviluppo. L'unica accortezza da utilizzare nell'adottare Eclipse come IDE primario per il proprio lavoro di programmazione è quella di dotarsi di hardware particolarmente performante. Il tool infatti non è esattamente un campione di leggerezza

Directory: /Eclipse

DEVC++ 4.9.9.2

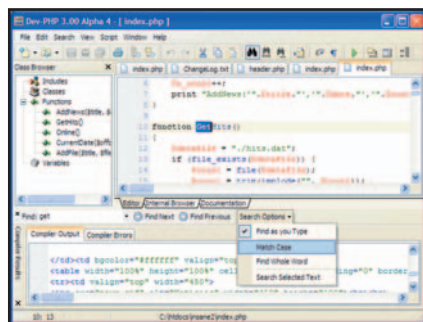
L'IDE PIÙ USATO DAI PROGRAMMATORI C++

Dev C++ è ormai famosissimo. Se siete dei programmatori C++ probabilmente ne apprezzerete la leggerezza. Se vi state avvicinando adesso al C++ troverete molto comode le sue funzioni di Highlithing e Code Complexion. L'ide supporta direttamente il compilatore MingGW, ma può essere facilmente integrato anche in altri compilatori. E' molto interessante usarlo in congiunzione con le wxWidgets, in questo senso infatti Dev C++ diventa persino un ambiente RAD, tale da consentire di disegnare le interfacce direttamente dall'ambiente di programmazione e gestirle poi con il classico modello ad eventi.

Si tratta di un progetto OpenSource, e come tali non ha costi da sostenere, è altamente integrato con il compilatore MingGW ma è utilizzabile anche con il framework .NET.
Senza dubbio un indispensabile da tenere a portata di mano sia quando si sviluppano progetti complessi, sia quando si ha a che fare con progetti di piccole dimensioni

Directory: /devcpp

DEVPHP

UN OTTIMO IDE PER PHP

Si avvia ormai ad essere un progetto maturo questo DevPHP. Un IDE completo di debugger per i progettisti di applicazioni Web che fanno di questo linguaggio il proprio cavallo di battaglia. La sua forza sta nell'esporre funzioni anche avanzate in modo del tutto originale. Certo non ha ancora la complessità ad esempio di ZEND, ma è dotato di tutte quelle caratteristiche essenziali che ne fanno uno strumento decisamente utile a chiunque desideri sviluppare applicazioni in PHP.

DevPHP rappresenta una delle poche opzioni praticabili per quanto riguarda PHP se non si vogliono spendere quantità industriali di quattrini per ricorrere a IDE commerciali.
È un progetto OpenSource ma gode di ottime qualità e offre funzionalità avanzate. Senza dubbio uno strumento utile per ogni programmatore PHP

Directory: /devPHP

DOTPROJECT 2.0

UNA WEB APPLICATION PER IL CONTROLLO DEL FLUSSO DI LAVORO

Se avete necessità di controllare il processo lavorativo di piccoli gruppi, questo tool è quello che fa per voi. Consente di schedulare le attività, monitorare lo stato d'avanzamento di un progetto, gestire le risorse e i contatti. La Web application si configura come un repository centralizzato di informazioni, tale che chiunque appartenga alla Intranet vi possa accedere con diversi livelli di protezione a seconda del ruolo per cui è stato registrato.

Nel complesso un ottimo tool, non completo come Tutos che nel suo genere rappresenta il top della gamma, ma sicuramente ben fatto, meno complesso e con un'interfaccia più intuitiva

Directory: /dotproject

FREETTS 1.2.1

UN SINTETIZZATORE VOCALE SCRITTO INTERAMENTE IN JAVA

La sintesi vocale rappresenta uno dei futuri traguardi della programmazioni. Applicazioni che interagiscono con l'utente non solo per mezzo dei metodi di input/output tradizionali tastiera/stampante ma anche tramite metodi innovativi quali la voce utilizzata sia per impartire comandi sia per sintetizzare risposte. È in quest'ottica che si inserisce FreeTTS grazie al quale sarete in grado di farvi leggere dal computer un testo e questo non perché esso sia stato preventivamente registrato, ma perché il sintetizzatore riconosce le parole,

riesce a leggerle e a pronunciarle in modo adeguato. Di FreeTTS abbiamo parlato abbondantemente anche nei numeri 90 e 91 di ioProgrammo in cui appunto si descriveva la costruzione di una completa applicazione di sintesi vocale in Java che utilizzava appunto FreeTTS

Directory: /freetts

HIBERNATE 3.1.1

IL TOOL DI PERSISTENZA CHE STA CAMBIANDO IL MODO DI PENSARE AI DB

Chi lavora con i database è abituato a pensare in termini di relazioni. I programmatori viceversa sono abituati a pensare in termini di oggetti. Ovviamente quando in un programma si fa uso di un database, i dati rimangono slegati dal contesto del programma, infatti difficilmente sono rappresentabili da oggetti. Hibernate è un framework di persistenza che prima di ogni cosa mette a disposizione del programmatore un sistema di mapping fra dati e struttura programmatica, rendendo gli elementi del database accessibili come oggetto. Si occupa poi di gestire ogni altro aspetto della programmazione, come il salvataggio e il recupero dei dati oppure delle sessioni. E' interessante anche notare come con Hibernate esista un elevato grado di disaccoppiamento fra il database e le classi che lo gestiscono. Non è ancora un'implementazione completa di IOC ma ci si avvicina abbastanza. In questo numero presentiamo anche NHibernate, ovvero la versione .NET dello stesso progetto

Directory:/Hibernate

HSQLDB 1.8

IL DATABASE LEGGERO E EFFICIENTE

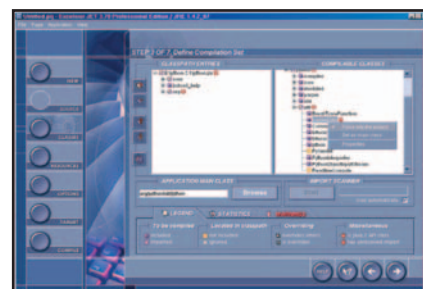
HSQLDB è diventato da poco uno standard ed integrato in OpenOffice. I Windowssiani possono considerarlo come una sorta di Microsoft Access. Si tratta di uno strumento davvero assolutamente leggero, utilizzabile sia in modalità standalone che in modalità server. Il suo utilizzo nasce per essere associato a Java ma con poche accortezza è possibile utilizzarlo anche con linguaggi diversi da Java.

Il principio è sempre lo stesso, tutti i dati vengono inseriti in unico file. HSQL può essere eseguito in modo standalone o come server. La differenza con Access sta nelle prestazioni. A differenza del secondo, HSQL è un motore de DB dalle capacità prestazionali di tutto rispetto, in grado di gestire senza sforzo quantità di dati considerevoli

Directory:/hsql

J2SE 5.0 UPDATE 6

LA SESTA RELEASE DEL COMPILATORE PER PROGRAMMARE IN JAVA

Nel mese di Gennaio i progetti Java disponibili su SourceForge, una delle più grandi community per lo sviluppo al mondo, hanno per la prima volta superato quelli relativi a C++, segno di un'innegabile successo del linguaggio progettato da Sun.

Java è ormai diventato un linguaggio affidabile ed elegante, tanto che i suoi utilizzatori crescono di giorno in giorno. Il compilatore principe per i progetti Java è questo J2SE sviluppato da Sun, giunto alla sua versione 5.0 e all'upgrade 6 che corregge alcuni piccoli bug e aumenta ancora considerevolmente l'affidabilità del compilatore. In questa versione è com-

Ajax.NET

IL COMPONENTE PER USARE AJAX CON VISUAL STUDIO

Di Ajax ne abbiamo parlato abbondantemente nei numeri scorsi di ioProgrammo e continuiamo a parlarne in questo numero con il bell'articolo di Fabio Cozzolino.

Ajax è una di quelle tecnologie che sta radicalmente cambiando il modo di programmare il Web. Sostanzialmente consente di effettuare un reload di piccoli pezzi di pagina senza dover ricaricare l'intera pagina.

È concettualmente diverso da un iFrame ad esempio, perché non si tratta di un componente separato dalla pagina stessa, ma di un elemento incorporato che viene modificato a runtime sulla base della tecnologia Ajax, del DOM e di JavaScript. Tutto questo consente di velocizzare enormemente il tempo di load di una pagina web, e di creare applicazioni internet, la cui interfaccia ha un comportamento simile a quella delle applicazioni standalone.

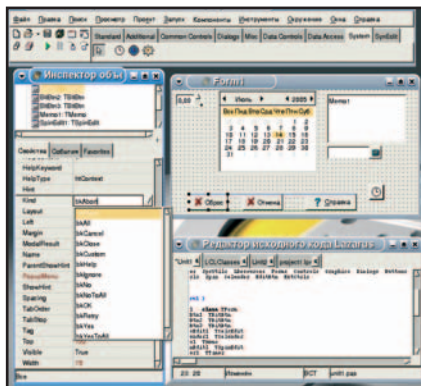
Directory: /Ajax

preso anche NetBeans, ambiente di programmazione per Java certamente non leggero ma molto ben strutturato. NetBeans accellera notevolmente il processo di sviluppo utilizzando Java, l'ambiente è dotato di tutte le funzionalità essenziali, e si preconfigura per certi versi anche come un RAD avanzato
Directory:/ J2se

LUCENE 1.4.3
UN SEGUGIO PER I DOCUMENTI

Di Lucene abbiamo parlato a lungo in qualche numero precedente di ioProgrammo. Si tratta di una libreria che consente l'indicizzazione di file di ogni tipo presenti nell'hard disk, al fine di poterli ricercare tramite un'applicazione. Lo scopo è quello di realizzare software molto vicini a Google Desktop Search. La libreria è realizzata in modo sapiente e sia l'indicizzazione che la ricerca sono risultate molto veloci. Presentiamo sia la versione per Java che la versione per l'ambiente .NET. Su Lucene si basano alcuni progetti OpenSource che stanno rapidamente conquistando i favori degli utenti. In particolare Beagle è un tool di ricerca associato al progetto Gnome e che sta diventando un riferimento per quanto riguarda l'indicizzazione di documenti sull'Hard Disk.
Directory:/ Lucene

LAZARUS 0.9.10
IL CLONE DI DELPHI

Chi ha usato almeno una volta Delphi ne è sicuramente rimasto affascinato. Borland è stata la prima a creare un ambiente di programma zione con interfaccia RAD facile ed intuitivo

Grazie all'avvento di Delphi si è segnato un punto di svolta nelle tecniche programmative. Il linguaggio Object Pascal contenuto al suo interno non ha poi per motivi oscuri raggiunto il successo sperato, rimane tuttavia un punto di riferimento per chiunqe vogli avventurarsi nel magico mondo della programmazione.
Si tratta probabilmente del più sofisticato IDE oggigiorno in commercio. Peccato che il costo sia decisamente elevato e pertanto non facilmente accessibile al programmatore singolo o alla piccola software House. Per costoro c'è Lazarus il clone OpenSource di Delphi. Certo è ben lontano dall'essere la soluzione sofisticata offerta dall'ambiente di Borland, ma rappresenta un punto di ingresso a basso costo per coloro che si affacciano a questo mondo
Directory:/ Lazarus

NHIBERNATE
COME HIBERNATE MA IN AMBIENTE .NET

Neanche .NET sfugge alla logica secondo cui i database relazionali sono difficilmente rappresentabili come oggetti. Ed ecco che arriva NHibernate che seguendo lo stesso paragdima di Hibernate garantisce il mapping fra oggetti ed elementi di un database relazionale superando i limiti imposti dalle due diverse tecnologie. Il tool è particolarmente interessante, sia per l'elevato grado di disaccoppiamento che offre tra gli elementi del database e le classi che lo gestiscono, sia per la capacità di trasformare dati tipicamente trattati in modo relazionale in classi ed oggetti
Directory:/ Nhibernate

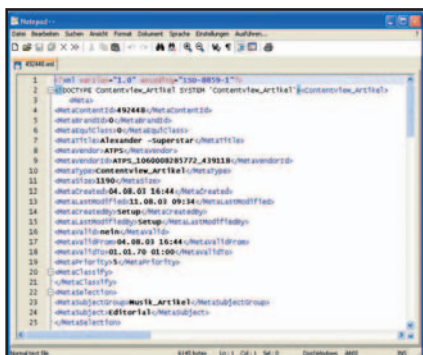
MYSQL 5.0.18
LA NUOVA VERSIONE DEL DATABASE LEADER SUL WEV

MySQL alimenta una percentuale vicina al 90% delle Web Application. Si tratta di un database estremamente veloce che ha trovato sul web la sua collocazione ideale grazie all'alta integrazione con PHP, alla leggerezza, alla facilità di installazione.
Il suo ambiente ideale è quello del Web. Di fatto la maggior parte delle applicazioni disponibili su Internet e che ha che fare in qualche modo con l'uso di un database, si interfaccia in una certa misura con Mysql. Fino a qualche tempo fa aveva il limite di non disporre di interfacce grafiche di gestione, attualmente questo limite è stato abbondantemente superato, con l'avvento di strumenti sofisticati.

Recentemente MySQL ha trovato una nuova maturità grazie all'introduzione di funzionalità come le transazioni, la ricerca full text e nelle ultime versioni anche le stored procedure e i trigger. Si tratta di un prodotto ormai completo che ha ben reagito agli attacchi sferrati dalla concorrenza proprio sul piano del web, che non ha perso posizioni nel suo mercato di riferimento, e che anzi ne sta guadagnando sul terreno ad esso meno congegnale delle applicazioni standalone
Directory:/ MySQL

NOTEPAD ++
L'EDITOR EVOLUTO MA LEGGERO

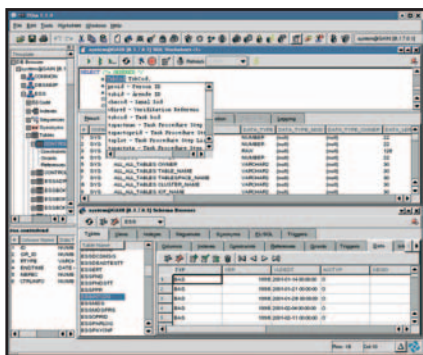
Il notepad è stato per lungo tempo lo strumento più amato dai programmatori. Leggero, veloce, essenziale, privo di ogni fronzoli consente di aprire e modificare un file di testo in pochi attimi. Rimane ancora oggi un software particolarmente amato dai programmatori, che lo utilizzano per una gran varietà di scopi. Anche se talvolta non si può rinunciare alla comodità di un IDE, il notepad rimane comunque un'ottima soluzione quando si devono apportare modifiche rapide, di poche linee di codice ad una qualunque applicazione. Nasce comunque l'esigenza di un

notepad migliorato quando si devono ad esempio aprire file di grandi dimensioni, oppure quando si necessita della funzione di sintax highlighting.

Notepad ++ supera questi limiti e si propone come un notepad avanzato, semplice come l'illustre progenitore ma in grado di aprire file di qualunque dimensione e di gestire qualunque linguaggio di programmazione

Directory:/ Notepadpp

PGADMIN III

L'INTERFACCIA DI GESTIONE DI POSTGRESSQL

PostgresSQL è un database eccezionalmente potente, forse il più completo oggi disponibile. Il suo tallone d'achille risiede probabilmente nella difficoltà del linguaggio che ne sottintende alla gestione.

Per evitare lo scontro con una sintassi non sempre chiara c'è pgAdmin, una comoda interfaccia grafica attraverso la quale si possono gestire tutte le funzionalità di un server Postgres in modo semplice ed ottimale. Postgtes rimane un database incredibilmente evoluto, che fa dell'estendibilità uno dei suoi punti di forza grazie al suo linguaggio interno

Directory:/PGAdmin

POSTGRES 8.1.2

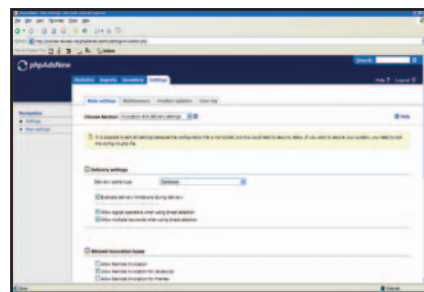
IL DATABASE COMPLETO, VELOCE ED EFFICIENTE

Ci ha lasciato sbalordito questo database quando qualche numero fa di ioProgrammo ne abbiamo effettuato una prova e pubblicato un articolo sulle caratteristiche più importanti. Estremamente flessibile e veloce, contiene alcune funzionalità che non sono tipicamente esposte neanche in database commerciali dal costo di parecchie migliaia di euro. Si va dalle classiche stored procedure, ai trigger, alla programmazione modulare a mezzo di comodi plugin. Nelle versioni della serie 8 è stato aggiunto un comodo installer grafico che lo rende adatto ad essere installato in pochi passi anche in ambiente windows.

Directory:/postgres

PHPADSNEW 2.0.7

UN BANNER SERVER MOLTO POTENTE

Se avete un sito web e avete provato a vendere qualche banner a un cliente vi sarete trovati nella necessità di contare il numero di impressioni, il numero di click e probabilmente di offrire una reportistica al cliente.

Se poi il vostro sito espone più di un banner, sicuramente la cosa si sarà complicata ulteriormente.
phpAdsNew è un banner server, consente di tenere traccia del numero di esposizioni, del numero di click, della rotazione, e di una serie di altri parametri professionali tipici di software di Advertising

Directory:/ phpAdsNew

PHP 5.1.2

LA NUOVA VERSIONE DEL LINGUAGGIO PRINCIPE PER IL WEB

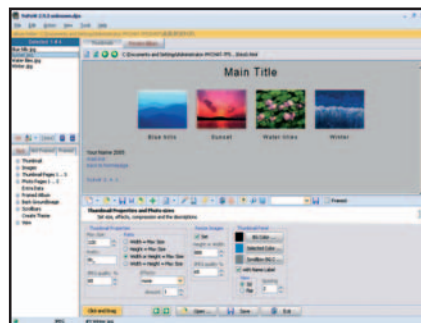
PHP è probabilmente il linguaggio tramite il quale sono state sviluppate la maggior parte delle applicazioni che fanno girare oggi internet. Dalla sua parte la curva di apprendimento molto rapida, la completezza delle funzioni esposte, la licenza opensource, la possibilità di programmare in modo procedurale oppure ad oggetti.

Questa nuova versione 5.1.2 corregge alcuni bug della versione precedenti, esporta un miglior modello ad oggetti, implementa l'interfaccia PDO tramite la quale è possibile sganciarsi dal database utilizzato e sviluppare non tenendo conto del backend sottostante

Directory:/PHP

PHPBB 2.0.19

UNO STANDARD FRA I FORUM

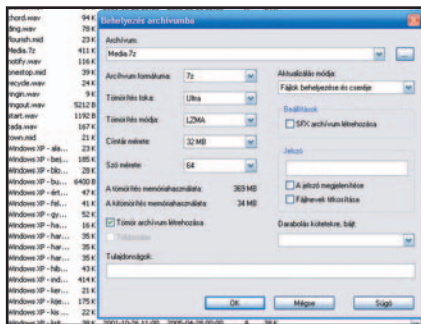
Non c'è community che non abbia un forum! PHPbb è un'applicazione ne implementa uno ottimo e completo. Consente la creazione degli argomenti, la moderazione, il reply con visualizzazione in modalità discussione, il punteggio per argomento, l'antispam e una serie di altre operazioni piuttosto utili per un amministratore di un forum, cosi' come per l'utilizzatore.

Unico punto a sfavore: la sicurezza. Essendo un'applicazione molto diffuso, quasi uno standard, gli hacker passano moltissimo tempo alla caccia di bug o piccoli errori di program-

mazione per comprometterne il funzionamento. Tuttavia il team di sviluppo reagisce abbastanza in fretta rendendo disponibile quasi immediatamente patch risolutive

Directory:/ PHPbb

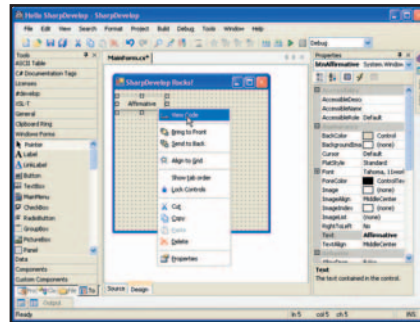
SEVENZIP 4.3.2

UN SOFTWARE DI COMPRESSIONE ECCEZIONALE

All'inizio fu Winzip, poi venne l'era dei cloni! SevenZIP ha superato l'illustre progenitore. L'algoritmo di compressione utilizzato da SevenZip in molti casi fornisce risultati superiori a qualunque aspettativa.
Inoltre SevenZip è completamente OpenSource, sono disponibili i sorgenti e questo rappresenta un'ottima occasione per i più curiosi di capire come funzionano gli algoritmi che stanno alla base delle tecniche di compressione

Directory:/ Sevenzip

SHARPDEVELOP 2.2.0.92

L'ALTERNATIVA A VISUAL STUDIO

Con l'avvento dei tool Express di Microsoft, sembrerebbe che l'utilità di questo prodotto sia destinata a decadere. Di fatto SharpDevelop è stata fino a ieri l'ancora di salvezza per quanti desideravano programmare in C# ma non potevano permettersi di sostenere i costi abbastanza elevati di Visual Studio 2003. Proprio per la sua enorme diffusione il prodotto continua e continuerà ad essere sviluppato, per quanti essendosi abituati ad usarlo non hanno nessuna intenzione di cambiare in favore del "tool ufficiale". In ogni caso SharpDevelop rimane un ambiente interessante che offre alcune funzionalità di rilievo e alcuni wizard che semplificano notevolmente le operazioni di programmazione giornaliere.

Inolte SharpDevelop è leggero ed essenziale e rappresenta ancora adesso una buona soluzione per chi non dispone di macchine dalle enormi risorse

Directory:/ SharpDevelop

SPHINX4

IL RICONOSCITORE DI PAROLE

Abbiamo già parlato di FreeTTS come un motore in grado di riprodurre un testo e trasformarlo in un output vocale. Non abbiamo detto però qual'è il tool che consente di riconoscere una parola, stabilire il giusto accento e la giusta pronuncia, questo tool è Sphinx4.
La versione presentata è la 1.0 beta, non tutte le parole sono riconosciute in modo ottimale, il legame con la lingua italia è scarso, ma se volete iniziare a provare come funzionano tool di questo genere sicuramente è un ottimo inizio

Directory:/ Sphinx4

SPRING 1.2.6

IL FRAMEWORK DEI FRAMEWORK

Ne parliamo abbondantemente nell'articolo di Roberto Sidoti in questo stesso numero di ioProgrammo. Spring è un Framework che riunisce sotto un unico cappello una serie di strumenti già esistenti mettendoli in grado di comunicare in modo corretto fra loro. Inoltre Spring implementa il pattern Ioc, inversion of control che garantisce un alto grado di disaccoppiamento e una maggiore manutebilità del codice

Directory: /Spring

STRUTS 1.2.8

IL FRAMEWORK CHE GARANTISCE L'MVC

L'avvento di Struts ha segnato in modo permanente il modo di intendere il web per quanto riguarda la programmazione con le JSP. Struts implementa totalmente il pattern Model, View, Controller, che prevede una stratificazione di un'applicazione su tre diversi livelli. Il modello implementa la logica di business dell'applicazione, la view si occupa di gestire l'output e il controller lavora sul flusso d'esecuzione del programma. Sviluppare una web application in questo modo la rende particolarmente facile da gestire e manutenere

Directory:/ Struts

SUBVERSION 1.3.0

IL SOFTWARE EMERGENTE PER IL CONTROLLO DI VERSIONE

Quante volte vi è capitato di modificare profondamente un software per poi dover tornare a una versione precedente a causa di qualche malfunzionamento? Oppure nello sviluppo in team quante volte vi è capitato che un collega maldestro abbia sovrascritto il vostro lavoro?
Subversion è un software per il controllo di revisione che garantisce di poter lavorare in gruppo, mantenere un log dei cambiamenti e una serie di altre opzioni che sono necessarie per seguire in maniera omogeneo lo sviluppo di un'applicazione. Per molti è l'erede del CVS che fino a poco tempo fa svolgeva egregiamente questi compiti

Directory:/ subversion

TOMCAT 5.5.15

L'APPLICATION SERVER PER LE JSP

Se sviluppate in JSP o avete intenzione di farlo non potete fare a meno di utilizzare Tomcat, il principale application server che vi consente di lavorare con le servlet e le JSP.

Si tratta di un prodotto ormai maturo che garantisce larga affidabilità e che abbastanza diffuso da incontrare il favore di una community larga e competente abbastanza propensa a condividere documentazione ed

esperienze per poter lavorare in modo efficiente con questo server
Directory:/ Tomcat

TUTOS 1.2.2

IL LEADER DEI SOFTWARE DI WORKGROUP

Se vi trovate a dover gestire un gruppo di lavoro o un progetto o più gruppi che lavorano su più progetti, tutos è il software che fa per voi. Si tratta di una web application che consente di automatizzare le scadenze ,lo stato d'esecuzione, le risorse, tutti i parametri tipici necessari all'organizzazione delle procedure gestionali che consentono di ottimizzare il flusso del lavoro. Tutos esporta una serie di funzionalità fuori dal comune che vanno dalla creazione dei diarammi di Gantt fino alla gestione e condivisione dell'indirizzario.
Assolutamente un must per chi si occupa di gestione della produzione
Directory:/ tutos

URLREWRITING .NET

PER REINDIRIZZARE UN URL IN MODO PERSONALIZZATO

Chi è abituato a lavorare con Apache conosce già l'utilità del modulo rewrite che consente di riscrivere un indirizzo generato dal client in un modo conforme alle proprie esigenze, cosi che per esempio l'indirizzo http://www.nomesito.it?nomeparametro=nomevalore possa essere riscritto automaticamente come http://www.nomesito.it/miapagina nascondendo gli altri valori al client. UrlRewriting è un clone del modulo rewrite in ambiente .NET che assolve appunto a questa funzione. Utilizzare questo modulo consente da un lato di innalzare la sicurezza delle proprie applicazioni, dall'altro consente di utilizzare una sintassi elegante che evita di dover scrivere centinaia di parametri per il passaggio delle variabili.
Directory:/urlrewriter

VELOCITY 1.4

PER CREARE TEMPLATE IN JAVA

Chi lavora in JSP sa quanto sia noioso miscelare in uno stesso file tag di output con codice java, allo stesso modo scrivere applicazioni Java che debbano generare un output testuale può diventare un'operazione delicata.
Velocity consente di generare dei template che vengono interpretati da un motore che sostituisce solo le parti dinamiche del template con variabili generate dall'applicazione. In questo modo si riesce a separare la logica di programmazione da quella di presentazione garantendo uno sviluppo semplificato al team di lavoro
Directory:/Velocity

WORDPRESS 2.0

IL PRINCIPE DEI BLOG

Il primo ed il più leggero software di blog ad aver fatto la sua comparsa sul mercato. Wordpress si differenzia per la sua leggerezza e per la scarsità di moduli presenti nel software di base. Tuttavia esiste una quantità straordinaria di plugin che possono arricchirlo così che il software fa della scalabilità uno dei suoi punti di forza.

D'altra parte la sua longevità dimostra quanto questo approccio sia efficace.
Directory:/ Wordpress

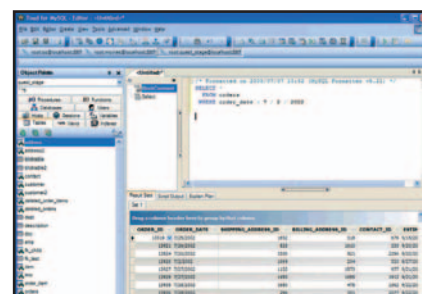
SQL BUDDY 0.70

UN EDITOR SQL COMPLETO

Un ottimo editor SQL che si sostituisce egregiamente al Query Analizer di Microsoft e che consente di connettersi agevolmente a database a basso costo come MSDE
Directory:/ SqlBuddy

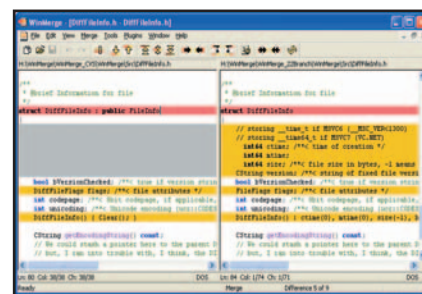
TOAD FOR MYSQL

LINEA DI COMANDO ADDIO

Toad è un tool grafico per l'amministrazione di database. In questo numero vi presentiamo la versione freeware per Mysql. In realtà essendo questa una preview, il software ha una scadenza di 60 giorni oltre la quale è necessario effettuare una nuova installazione.

In ogni caso Toad è un ottimo Frontend dotato di un Sql Editor, uno schema browser, un utilissimo gestore delle sessioni e altri tool interessanti come per esempio un comparatore di Database che vi individua tutte le differenze fra un database e un altro.
Directory:/ Toad

WINMERGE 2.4.4

EFFETTUA UNA COMPARAZIONE DI DUE FILE DI TESTO ANCHE DI GRANDI DIMENSIONI

WinMerge è un software Open Source che consente di effettuare il merging e il compare di due file di testo.

È utilissimo quando si vogliono comparare due diverse versioni di un file
Directory:/ winmerge0

IDEE SUL MASSIMO COMUN DIVISORE

ESAMINANDO IL SEMPLICE PROBLEMA MATEMATICO SUL CALCOLO DEL MASSIMO COMUN DIVISORE, RIPORTIAMO LA NOSTRA LENTE DI INGRANDIMENTO SUL CONCETTO DI ALGORITMO IN GENERALE

Intorno al 1950 la parola algoritmo era spesso associata con "l'algoritmo di Euclide". Ci viene richiamato alla memoria da Donald Knuth nel suo importante trattato: "The art of computer programming". Ricordo come fosse ieri il primo algoritmo appreso a scuola, era proprio quello di Euclide per il calcolo del massimo comun divisore (figura 3). Si tratta quindi del più significativo

Fig. 3: **"Flow Chart per il metodo di Euclide per il calcolo del MCD"**

prototipo di algoritmo ed è praticamente il punto di partenza di molti altri metodi. È fondamentale nel trattamento di equazioni indeterminate, ad esempio con il metodo dell'identità di Bezout, permette di comparare due rapporti per stabilire se sono uguali. Ed ancora è alla base di un importante algoritmo, realizzato da Sturm ed usato in molti ambiti tra cui i controlli automatici, per stabilire il numero di radici reali di una equazione algebrica in forma polinomiale. Inoltre, la realizzazione del metodo ha storicamente sollevato il problema di rappresentare in qualche modo gli algoritmi; sono così stati introdotti i flow chart, risultati però inadeguati per la programmazione strutturata, che oggi rappresenta il paradigma di sviluppo. Insomma, abbiamo a che fare con uno dei mattoni delle fondamenta della casa chiamata "programmazione".

MASSIMO COMUN DIVISORE

Il concetto matematico che esprime tale operazione è semplice; il nome è auto esplicativo. Dati due numeri A e B il massimo comun divisore (da ora in avanti MCD) è un numero divisore di entrambi (comune) i numeri A e B. Tra tutti i possibili divisori comuni si scegli il maggiore (massimo). Facciamo subito qualche esempio. Sia A=30 e B=6. In tal caso il MCD è uno dei due numeri: 6. Infatti 6 è divisore sia di 30 sia dello stesso 6. Si può facilmente verificare che vi sono divisori più grandi di 6. Secondo esempio: si considerino A=25 e B=15. Qui il MCD è 5. In forma organica tale operazione fu introdotta da Euclide. Le prime tracce del MCD che, ovviamente non aveva tale nome, si riscontrano con Aristarco di Samo che nel suo esemplare trattato astronomico: "Sulle Dimensioni e Distanze del Sole e della Luna" calcolava il rapporto 71.755.875:61.735.500 sostituendolo con 43:37; ed ancora sostituì 7921:4050 con 88:45. Semplificazioni che presupponevano il calcolo di un comune divisore. Anche Archimede nei sui calcoli di π greco sostituì dei rapporti tra interi con numeri più piccoli servendosi di comuni divisori. In effetti, il MCD ha tra le sue applicazioni quella di semplificare rapporti tra interi. Infatti per una frazione, dividendo numeratore e denominatore per il MCD il rapporto risulta semplificato. Con riferimento al secondo esempio il rapporto 15:25 è equivalente a 3:5. Esistono comunque applicazioni più sofisticate.

ALGORITMO DI EUCLIDE

Il matematico greco fu il primo che si occupò della questione. Come espresso nel libro Ele-

menti, egli affrontò il problema da un punto di vista geometrico. La grande intuizione di Euclide, una volta considerati i numeri come misure, fu di esprimere ognuno di essi come la composizione di unità più piccole, e di considerare la più grande possibile delle unità a disposizione. Lo studio si compone di due proposizioni. La prima mostra la soluzione degenere 1, nel caso di numeri per i quali uno dei due non è primo del secondo. Tale situazione si ha ovviamene quando i due numeri sono entrambi primi, da cui si deduce facilmente che il più grande divisore comune sia l'unità. Ma anche quando i due insiemi di primi che compongono i numeri non hanno elementi in comune, ossia la loro intersezione è nulla. Per capire la situazione si consideri i due numeri 21 e 10. Entrambi non sono primi, inoltre nessuno dei due è primo dell'altro, ossia i due insiemi di primi in cui si possono scomporre i due numeri sono {3,7} per 21 e {2,5} per 10 che evidentemente non hanno elementi in comune. La seconda proposizione mostra il caso di MCD come numero diverso da 1, che si ha nella circostanza opposta alla precedente; quando un numero è primo dell'altro. Nella proposizione 1 si considerano due numeri (misure) e si sottrae il più piccolo dal più grande. Lasciato il più grande si focalizza l'attenzione tra il restante numero e il risultato ottenuto. A questi due numeri si applica lo stesso trattamento, ovvero si sottraggono. Ancora una volta il più grande è scartato. Il procedimento è iterato finché il risultato ottenuto è 1. Se si arriva a tale risultato vuol dire che ci troviamo nel primo caso e quindi il MCD è 1. Euclide considerò due misure, ossia due segmenti AB e CD. Sottrasse il più piccolo dal più grande, scartato il segmento più grande, il segmento ottenuto come risultato veniva nuovamente esaminato insieme al rimanente. Iterando il procedimento così come mostrato in figura 1 si sviluppò il metodo.

Con la seconda proposizione Euclide contemplò il caso più generale di misure che avessero in comune segmenti diversi dall'unità. Tale evenienza si presenta ogni qual volta una delle due misure non è un primo dell'altro. Il procedimento è identico al precedente con eccezion fatta per il criterio di terminazione che in questa seconda proposizione è dato dalla eguaglianza tra il risulta-

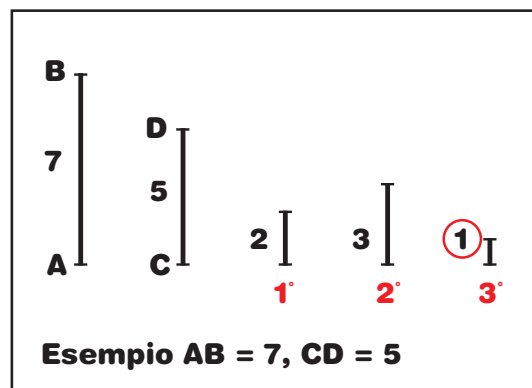



Fig. 1: "Le diverse iterazioni che mostrano la prima proposizione del metodo di Euclide per il calcolo del MCD"

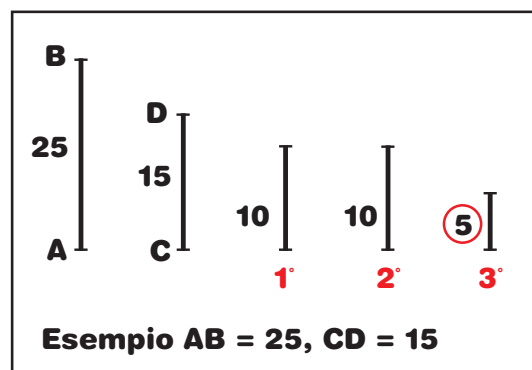



Fig. 2: "Le diverse iterazioni che mostrano il procedimento geometrico della seconda proposizione del metodo di Euclide"

DAL FLOW CHART ALLA PSEUDO CODIFICA

Agli albori della programmazione di elaboratori, il personale addetto al compito era molto specializzato nonché raro. Con gli anni all'aumentare dei computer è necessariamente aumentata la necessità di elaborazione dei dati. Sono così sorti i primi modelli per sviluppare i programmi in modo il più possibilmente condivisibile. Un modo per concretizzare facilmente i passi dell'algoritmo. Il metodo che da subito si è imposto è il flow chart, da noi conosciuto come diagramma di flusso. Si tratta di uno schema (come in figura 1), per la rappresentazione dell'algoritmo. È risultato molto efficiente per linguaggi non strutturati come il Basic. Ha mostrato invece limiti con la diffusione di linguaggi come Pascal, C++ e Java che per loro natura strutturata non si prestavano a tradurre tutti gli algoritmi sviluppati con i flow chart. Così per potere usare i diagrammi di flusso per linguaggi strutturati si sono introdotte alcune regole restrittive nello sviluppo di tali schemi. Negli stessi periodi sono stati sviluppati altri metodi di descrizione di un algoritmo come i diagrammi a scatola in cui le varie strutture di controllo (sequenza, selezione e ciclo) sono scatole che possono tra loro risultare in sequenza o disposte una dentro l'altra, come per le scatole cinesi. Ma anche questo metodo non ha trovato molti proseliti. La ragione sta nella naturalezza con cui si programma con i linguaggi strutturati che hanno reso di fatto inutile il passaggio intermedio dall'ideazione dell'algoritmo alla codifica. Ad ogni modo sono usati dei meta-linguaggi che con comandi codificati nella lingua di appartenenza non sono altro che una traduzione delle istruzioni del linguaggio stesso. Con tale pseudocodifica, ad esempio, l'istruzione di selezione semplice si traduce come:

```
Se  <condizione> allora <istruzione 1>
Altrimenti <istruzione 2>
```

Con l'introduzione di sempre più sofisticati costrutti sintattici come la programmazione orientata agli oggetti (OOP), si sono resi necessari nuovi metodi di modellizzazione. Sono così stati introdotti schemi come UML.

to della sottrazione e uno dei due numeri. Tale valore sarà proprio il MCD. In figura 2 è mostrata la sequenza riferita a due segmenti che hanno in comune un terzo segmento diverso dall'unità. L'algoritmo di Euclide è la sintesi delle due proposizioni mostrate. Nella sua implementazione si tiene conto di numeri indicati ovviamente con variabili. Ecco l'algoritmo, e poiché ha un rilevante valore storico mostriamo anche il flow char associato.
Codifichiamo il risultato ottenuto. In C++ si ottiene:

```
// Metodo della sottrazione
unsigned long mcd_sottr
                    (unsigned long aa, unsigned long bb)
{  unsigned long x,y;
   x=aa; y=bb;
   while (x!=y)
   {  if (x>y) x=x-y;
      else y=y-x;
   };
   return x;
};
```

I due parametri sono i numeri di cui si vuole calcolare MCD, subito assegnati (per continuità con il flow char) alle due variabili x e y. Il codice presenta un'ulteriore ottimizzazione. Non è stata usata la terza variabile c come differenza tra i due numeri. In fase di controllo il numero più grande verrà aggiornato come sottrazione dal secondo. L'iterazione termina quando i due numeri sono uguali.

PROGRAMMAZIONE STRUTTURATA

I linguaggi che seguono tale concezione sono ideati in modo che i cicli possano essere disgiunti o al più innestati. Non è possibile che all'interno di un ciclo ci sia la chiusura di un secondo ciclo innescato fuori dal primo, eventualità che può ricorrere in linguaggi non strutturati come il Basic. In questi linguaggi esistono i salti incondizionati. I primi linguaggi strutturati che storicamente sono stati sviluppati devono i natali allo studioso Niclaus Wirth è sono Pascal e Modula 2. Altri linguaggi molto conosciuti che seguono la stessa filosofia sono C++ e Java. Si tratta comunque della famiglia dei linguaggi imperativi.

ALTRE SOLUZIONI

Un metodo analogo al precedente ma stilisticamente più elegante prevede la soluzione ricorsiva. Il criterio di uscita dalla funzione è l'eguaglianza dei due numeri. Nel caso di disuguaglianza, invece, si richiama la stessa funzione in due modi diversi a seconda se il primo numero (x) è maggiore del secondo (y) e viceversa. Nel primo caso si sottrae alla x la y e si passa tale valore come primo parametro, il secondo parametro rimane il numero più piccolo, appunto y. Nel secondo caso, analogamente, si passano sempre come parametri i due numeri più piccoli tra il risultato e i due numeri iniziali.

```
// Metodo ricorsivo della sottrazione
unsigned long mcd_ric
                    (unsigned long aa, unsigned long bb)
{  unsigned long x,y;
   x=aa; y=bb;
   if (x==y) return x;
   else
   {  if (x>y) return mcd_ric(x-y,y);
      else return mcd_ric(x,y-x);
   };
};
```

Per quanto più sintetica e formalmente più rispondente alla filosofia del metodo proposto da Euclide, l'efficienza di tale metodo è comparabile al caso precedente.
Si tratta di un modo diverso di codificare lo stesso metodo. Le soluzioni appena mostrate sono sicuramente semplici da comprendere, ma non risultano essere molto efficienti. La velocità con cui convergono al risultato non sono ottimali. Un modo per migliorare le prestazioni è stato mostrato sempre da Euclide.
Bisogna generare il nuovo numero da valutare anziché come sottrazione dei due precedenti come resto della loro divisione. In tal caso si converge molto più rapidamente alla soluzione. Inoltre, anche il codice risulta molto snello, non è necessario alcun controllo all'interno del ciclo poiché risulterà sempre y maggiore di x. Così è possibile assegnare al nuovo valore di x il vecchio di y e al nuovo di y il risultato dell'operazione di resto. Ecco il codice.

```
// Metodo del resto
unsigned long mcd_div
                    (unsigned long aa, unsigned long bb)
{  unsigned long x,y,z;
   x=aa; y=bb;
   while (y>0)
   {  z= x % y;
      x=y;
      y=z;
   };
   return x;
};
```

COMPARAZIONE TRA I METODI

Valutiamo in termini quantitativi i due metodi.

Per farlo introduciamo un criterio di confronto. In entrambi i metodi vi è un ciclo. Il numero di iterazioni è un elemento sicuramente valido per tale stima. Anche se a rigore il metodo con la sottrazione all'interno del ciclo esegue un confronto quindi è "leggermente" più lento. Nella misura dei tempi di computazione, conosciuta come analisi della complessità piccole differenze come quelle poco fa rilevate si considerano trascurabili.
Per contare il numero di cicli all'interno del while introduciamo una nuova variabile loops (appunto cicli). Verrà inizializzata a zero fuori dal ciclo e incrementata ad ogni iterazione. È dichiarata come variabile globale. Ecco come una delle due procedure può essere modificata. Analoghe correzioni si apportano alle altre.

```
unsigned long mcd_sottr
                    (unsigned long aa, unsigned long bb)
{  unsigned long x,y;
   x=aa; y=bb;
   loops=0; // Numero di cicli
   while (x!=y)
   {  loops++;
      if (x>y) x=x-y;
      else y=y-x;
   };
   cout<<"numero di cicli: "<<loops<<"\n";
   return x;
};
```

Una piccola precisazione per la correzione della routine ricorsiva. In questo caso loops va inizializzato fuori da ciclo e incrementato ogniqualvolta si accede alla funzione. Infatti, non ci sono cicli ma soltanto una serie di chiamate ricorsive. L'analisi di loops nei vari casi ci conferma ciò che avevamo intuito. Ho testato i metodi su diverse coppie di numeri e il risultato dà sempre un migliore comportamento per mcd_div. Riporto il risultato di una delle verifiche. Con la coppia di numeri rispettivamente pari a 9348255 e 4500000 il MCD è 45. Il numero di cicli (numero di accessi alla procedura nel caso di mcd_ric) risulta: 180 per il metodo con la sottrazione, 181 per l'analogo metodo con implementazione ricorsiva e soltanto 8 per il metodo con la divisione. Insomma, una bella differenza che mostra inequivocabilmente la sostanziale migliore efficienza del metodo del resto.

CONCLUSIONI

Per me che collaboro dal lontano n. 6 di ioProgrammo è un'occasione importante vedere che si è arrivati con slancio alla pubblicazione del n. 100. Per l'occasione avevo inizialmente pensato di presentare metodi complessi e di frontiera. Riflettendo a fondo ho deciso tutto altro approccio, affrontare un argomento che mi permettesse di esprimere alcuni riferimenti storici che hanno segnato la programmazione durante gli anni. Si può dire che ioProgrammo abbia accompagnato lo sviluppo dell'informatica nel corso della storia recente che comunque è nata da breve tempo. I quasi 10 anni di vita della rivista sono significativi, considerando gli sviluppi che il mercato della programmazione ha subito in Italia e nel mondo, sia in relazione alle tecniche usate, sia in relazione all'emergere di nuovi problemi da risolvere per noi programmatori. In tal senso l'analisi degli algoritmi rappresenta il modo migliore per tenere allenata la mente anche difronte all'emergere di tecnologie che spesso tendono a nascondere le funzioni di basso livello.

Fabio Grimaldi

RICORSIONE

È un modo per programmare. Una funzione è ricorsiva quando al suo interno c'è una chiamata alla funzione stessa. Oppure se vi è una chiamata ad una funzione che al suo interno richiama la funzione iniziale. Ogni qual volta una funzione chiama se stessa il compilatore dovrà preoccuparsi di salvare i dati critici della funzione chiamante in un apposito record di attivazione. Cosicché quando si ritornerà alla funzione chiamante tali dati saranno disponibili. I record sono disposti secondo una metodologia LIFO (last in first out) ossia in uno stack (pila). Infatti, quando l'ultima funzione chiamata nella catena ricorsiva terminerà, il controllo dovrà passare alla sua chiamante, ovvero la penultima. Tale procedimento si sviluppa naturalmente con una pila. Gestione questa a carico del compilatore. Chi infatti decide di trasformare una funzione ricorsiva in iterativa deve utilizzare direttamente programmando uno stack.

ANALISI DELLA COMPLESSITÀ

Si tratta di una serie molto sofisticata di tecniche per la valutazione dell'efficienza di un algoritmo. Può essere spaziale se valuta la memoria usata e temporale se l'elemento analizzato è il tempo. In realtà ciò che interessa maggiormente è il tempo infatti se si parla di complessità la si identifica di fatto con quella temporale. La complessità viene misurata come una funzione dipendente dalle dimensioni della struttura (o strutture) dati e solitamente si indica con la lettera O. Se lavoriamo ad esempio con un vettore la complessità per ordinarlo con bubble sort è dell'ordine di n^2 poiché ci sono due cicli innescati che scorrono sul vettore, quindi O(n^2). Quando la complessità è un polinomio di n siamo comunque soddisfatti poiché i tempi di elaborazione sono "sopportabili". Ci preoccupiamo se la complessità è esponenziale, ovvero una funzione che ha n come esponente. In tal caso alcune volte l'algoritmo anche essendo corretto non è utilizzabile per i tempi di elaborazione elevati. È il caso di algoritmi in cui ad ogni passo l'elaborazione si sviluppa ad albero su più sotto problemi. Esiste un'altra classe interessante perché oggetto di ferventi studi teorici; si tratta dei problemi non polinomiali (NP) che per così dire forniscono tempi di computazione accettabili ma che non possono purtroppo essere espressi come polinomi di n.

Biblioteca

ON LINE

C-SHARP CORNER

Ricchissimo di informazioni su C# e su tutte le nuove tencnologie in particolare WinFX e LongHorn. Vi si trovano all'interno una serie di esempi molto utili per coloro che intendono lavorare con i nuovi linguaggi
http://www.c-sharpcorner.com/

ONDOTNET

Sito molto interessante con una serie di link, articoli e blog sulle tecnologie Microsoft. Ovviamente tutto riguarda le novità sui linguaggi di Microsoft e vengono presentati link e esempi utili che possono essere sfruttati per migliorare il proprio lavoro di programmazione quotidiana
http://www.ondotnet.com

JAVASCRIPT

Script, tutorial, documentazione su JavaScript il linguaggio principe per la scrittura di applicazioni dinamiche lato client. Con tutti gli esempi che ne coinvolgono l'uso
http://javascript.internet.com

JAVA J2SE 1.5

La nuova versione del compilatore Sun per il linguaggio Java conta ormai quasi un anno di vita, eppure non tutte le applicazioni sono state ancora migrate alla nuova piattaforma. Questo sopratutto per le innovazione apportate dalla nuova release, innovazioni abbastanza corpose da giustificare un rilascio di una major version siglata appunto con il numero 5.

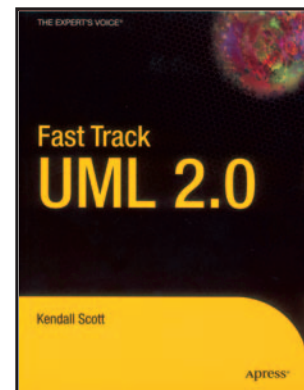
Il libro di Karsten Samaschke non è un manuale che illustra le differenze fra le varie versioni di Java, piuttosto è un riferimento certo per chi vuole imparare Java partendo direttamente con la versione più recente del compilatore. Si tratta di un libro intelligente, ben scritto e ben concepito. Si parte con la storia di Java per affrontare poi i costrutti di base del linguaggio e la filosofia di programmazione, per arrivare infine a trattare elementi più complessi quali l'integrazione con i database, e la gestione del networking. E' un libro adatto a chi inizia, ma anche chi ha una certa familiarità con la programmazione Java troverà spunti utili per arricchire la propria conoscenza.

Difficoltà: Bassa • **Autore:** Karsten Samashke • **Editore:** Apogeo • **ISBN:** 88-503-2332-8 • **Anno di pubblicazione:** 2005 • **Lingua:** Italiana • **Pagine:** 512 • **Prezzo:** € 39,00

FAST TRACK UML 2.0

UML è un linguaggio spesso troppo abbandonato a se stesso. Si tratta invece di uno strumento solido e importante che ogni programmatore dovrebbe utilizzare per progettare in modo certo il proprio lavoro.

Il libro è ben scritto e scorre via in modo piuttosto veloce. La lingua inglese è purtroppo un ostacolo per quanti non ne hanno padronanza, tuttavia lo stile è quello anglossassone preciso, pratico e puntuale. Non ci sono giri di parole ne considerazioni inutili, il libro mira diritto al sodo trattando gli argomenti di base con descrizioni chiare e precise. Si parte affrontando argomenti come le classi e le relazioni fra le classi per poi arrivare a discutere di template e di flussi. Si tratta di un libro scritto per utenti esperti che abbiano già dimestichezza almeno con un linguaggio di programmazione e che vogliano imparare a utilizzare uno strumento importante per la progettazione dei propri software.

Difficoltà: Alta • **Autore:** Kendall Scott • **Editore:** Apress • **ISBN:** 1-59059-320-0 • **Anno di pubblicazione:** 2004 • **Lingua:** Inglese • **Pagine:** 172 • **Prezzo:** $ 24,99

MAXIMIZING .NET PERFORMANCE

Il framework .NET ha sicuramente cambiato il modo di programmare di molti sviluppatori. Una serie di wizard e codice precostruito nasconde tutte quelle difficoltà dei linguaggi a basso livello che spesso fanno diminuire vertiginosamente la produttività. Paradossalmente queste facilitazioni si pagano in termini di controllo assoluto sul codice. A questo punto le vie da seguire sono due, o ci si fida di quello che hanno fatto i programmatori Microsoft o si interviene per eliminare quella serie di controlli o di routine che non ci servono realmente e che tuttavia diminuiscono le performance del software che stiamo scrivendo. E' di questa serie di problematiche che si occupa il libro "Maximizing

.NET performance" che con una serie di considerazioni ed esempi pratici ci porta a realizzare codice .NET con il massimo delle prestazioni possibili. Si tratta di un libro interessante che ciu porta all'interno di un mondo troppo spesso trascurato. Il libro è ben scritto, anche se la lingua può rappresentare un ostacolo difficile per chi non ha dimestichezza con l'inglese.

Difficoltà: Alta • **Autore:** Nick Wienholt • **Editore:** Apress • **ISBN:** 1-59059-141-0 • **Anno di pubblicazione:** 2004 • **Lingua:** Inglese • **Pagine:** 280 • **Prezzo:** € 44,00